Revisiting Diameter in Directed Graphs

Ben Bals
CWI, Amsterdam, The Netherlands
Vrije Universiteit Amsterdam, The Netherlands

Joakim Blikstad
Amsterdam, The Netherlands

Daniel Dadush
CWI, Amsterdam, The Netherlands

Yasamin Nazari
Vrije Universiteit Amsterdam, The Netherlands
CWI, Amsterdam, The Netherlands

Jonas Schmidt
Bocconi University, Milan, Italy

Abstract

The reachability diameter (ReachDiam) of a directed graph is the maximum distance over all pairs u, v where v is reachable from u. This notion is present in the definition of shortcut sets, and the name was recently coined in that context by Haeupler, Jiang, and Saranurak [SOSA 2026]. While this is a very natural notion of diameter in directed graphs, and especially DAGs, it is so far not computationally explored. Other definitions of diameter in directed graphs are either trivial (infinite) in graphs that are not strongly connected (e.g., the classical definition) or are non-trivial only in highly restrictive graph classes (e.g., Min-Diameter).

We initiate the problem of computing the (approximate) reachability diameter from a fine-grained complexity point of view. Under certain fine-grained assumptions, we prove that there is no algorithm in time $\mathcal{O}(n^{\omega-\varepsilon})$ that gives *any* approximation of ReachDiam in *weighted graphs*. Similarly, there is no algorithm with better than 2-approximation for *unweighted graphs* in this time. To supplement this, we provide algorithmic upper bounds that lead to additive approximation of ReachDiam for unweighted graphs. Hence, we establish a strong separation between the weighted and unweighted cases, which makes this type of diameter different in nature than other known notions.

Considering the hardness in general weighted graphs, we also study special graph classes and get small constant approximations for DAGs with bounded width or graphs with bounded treewidth. Interestingly, our techniques also lead to exact hopsets with hopbound 2 for bounded treewidth graphs. This and some of our upper bounds for general graphs show technical connections between approximating ReachDiam and computing shortcut sets and hopsets.

2012 ACM Subject Classification Theory of computation → Shortest paths

Keywords and phrases Graph algorithms, Diameter, Fine-grained complexity, Shortcut sets

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.59

Related Version *Full Version*: `https://arxiv.org/abs/2606.08217`

Funding *Yasamin Nazari*: This publication is part of the project "Dynamic graph algorithms: distances and clustering" with file number VI.Veni.232.038 of the NWO Talent Programme 2023 which is (partly) financed by the Dutch Research Council (NWO).

1 Introduction

Computing the diameter of a graph is a fundamental problem in fine-grained complexity that has been widely studied both from an upper bound and a lower bound perspective. While the problem is very well-understood in undirected graphs, there is still much that we do

not know for directed graphs. In particular, it is not even clear what the *right definition* of diameter for directed graphs should be, depending on the application. For example, existing definitions of the diameter become trivial in a DAG. In this work, we focus on the notion of the *reachability diameter* of a directed graph recently defined by [26]. Although this is a very natural definition, it has not yet been computationally explored. Our goal in this work is to characterize the fine-grained complexity of computing or approximating this notion.

▶ **Definition** (Reachability Diameter). *Let G be a directed graph. Let* $\text{ReachDiam}(G) := \max_{(u,v)\in \text{TC}(G)} d(u,v)$*, where* $\text{TC}(G)$ *is the transitive closure of G. In other words, we consider the maximum distance over all pairs $u, v \in V(G)$ where v is reachable from u.*

This definition is closely related to the notion of diameter used for shortcut sets. In particular, [26] used low diameter decompositions based on this notion of diameter for constructing shortcut sets. Informally, a shortcut set is a set of edges H added to a graph G such that the reachability diameter of $G \cup H$ is bounded by a given parameter β, called the hopbound. While shortcut sets and their distance preserving variants, hopsets, have found many applications and received significant attention recently, surprisingly *computing* this relevant notion of diameter has, to the best of our knowledge, not yet been studied.

In addition to fine-grained lower bounds, we will show additive approximations for this problem that lead to good enough estimates when this diameter is large. This is the relevant regime for shortcut sets in general directed graphs, in which our algorithms can be used to *verify* the diameter reduction procedure after adding shortcut edges. However, we see that getting constant approximate solutions in the other regimes turns out to be hard.

First, let us discuss how this definition compares to other directed diameter definitions and how they compare in the difficulty of computation.

Classical Diameter. Classically, the diameter of a directed graph is the largest shortest path distance over all pairs of vertices. This is infinite when the graph is not strongly connected. This diameter can be estimated in a straight-forward way. Consider the following folklore approach: Pick an arbitrary vertex u. Compute an inward and an outward Dijkstra. Return the largest distance from or to u. It is easy to show that this is a 2-approximation for the classical diameter in weighted directed graphs. This also correctly returns ∞ when the graph is not strongly-connected.

The state-of-the-art for this diameter [36] improves this to a 3/2-approximation by a more sophisticated argument that involves sampling a set of vertices and exploring a limited number of vertices from this set. In fact, for this notion of diameter, this approach provides similar guarantees on directed and undirected graphs.

With such approaches, if for a pair of vertices u, v either of $d(u,v)$ or $d(v,u)$ is infinite, the algorithm returns infinity, which is the correct answer for this notion of diameter. Hence for graphs that are not strongly connected, this value is easy to compute but not very insightful as we do not receive any information about the vertex pairs that have a finite distance. In such graphs, we would like to have a notion of distance for each pair in the relevant direction, namely the direction where those vertices are reachable. This is exactly what the reachability diameter captures.

Note that if the classical diameter is finite (i.e., in a strongly connected graph), then it is equivalent to the reachability diameter. Therefore reachability diameter can be seen as a generalization of the classical notion. In particular, classical diameter reduces to reachability diameter simply by first checking for strong connectivity.

At a high level, this is why we expect the reachability diameter to be more difficult to compute. We make this formal in Section 4 where, among other things, we prove that there exists no constant factor approximation of reachability diameter in weighted graphs in near-linear time unless well-known fine-grained conjectures fail.

Round-trip Diameter. The roundtrip diameter is another diameter notion that attempts to capture the maximum distance in both directions: [3] describes it as "the distance between two vertices u and v corresponds to going from u to v and back". Formally, the round-trip diameter is defined as the maximum over all pairs u, v of $d(u, v) + d(v, u)$. This notion also has the same limitation that it is only finite in strongly connected graphs.

Min-Diameter. The min-diameter (MinDiam) was introduced to offer a meaningful notion of diameter in a directed graph that is not strongly connected and has received considerable attention [3, 19, 15, 8]. The min-diameter of a directed graph is defined as the maximum over all vertices u, v of $\min(d(u, v), d(v, u))$.

The goal with this definition is somewhat similar to ours, as this value will be finite as long as there is reachability in one direction for each pair. Hence, this value is still meaningful for graphs that are not strongly connected. However, this definition is still quite restrictive. For example, a DAG has a finite min-diameter if and only if it represents a total order. Equivalently, such graphs will have the following structure: a single path plus extra edges along the path. For general directed graphs, the DAG of strongly connected components must have such a structure for the MinDiam to be finite. Therefore, unless a directed graph has this very specific structure, its MinDiam will again be infinite and not informative. Conceptually, this can also explain why MinDiam turns out to be easier to compute than ReachDiam. On DAGs, MinDiam can be 2-approximated in near-linear time [3], which we show is impossible for ReachDiam under fine-grained hypotheses in Section 4.

Note that reachability diameter can also be seen as a generalization of the min-diameter: In the more interesting finite case, MinDiam is exactly equivalent to ReachDiam. The infinite case is easy to detect in linear time by computing strongly connected components and then a topological sort.

Recently, [8] proposed a 3/2-approximation for MinDiam of sparse unweighted DAGs in subquadratic time. Also on weighted general directed graphs, with a more sophisticated approach, MinDiam can be approximated to a constant factor in near-linear time due to a recent result by Chechik and Zhang [15].

Weighted vs Unweighted Reachability Diameter. Another interesting fact about this new notion of diameter that we establish is a strong separation between approximating the weighted vs unweighted case that does not hold for the other definitions. In particular, we show that in weighted graphs assuming certain known fine-grained conjectures, we cannot get any multiplicative approximation in time $\mathcal{O}(n^{\omega-\varepsilon})$, whereas we can compute approximations for the unweighted case. This also hints at why reachability diameter is harder to compute than the other notions, as most known diameter estimation techniques work both for the unweighted and the weighted case (possibly with extra technical steps and analysis).

Difficulty of Applying Known Techniques for Computing the Reachability Diameter. For both the classical definition and for MinDiam, the known algorithms involve a step of sampling vertices, and computing the in- and out-reachability from each sampled vertex. This will give an estimate that for the classic diameter leads to a 2-approximation directly, and with some additional steps also leads to a constant-factor approximation for MinDiam or better approximations of the classical diameter.

For reachability diameter, this scheme has a fundamental limitation: the estimates obtained from the furthest distance of a set of sampled vertices may not give any useful information on ReachDiam. First consider the folklore approach for classical diameter that runs an inward and outward Dijkstra from a single vertex u. The analysis then resorts to triangle inequality to approximate any distance $d(a, b)$ with $d(a, u) + d(u, b)$. For ReachDiam however, if one of the latter distances is infinite, we gain no information about the true distance between a and b, not even if it is finite.

Intuitively, a large part of the graph may show a similar behavior: Consider any directed graph with n vertices. Say, we add another n source vertices, with a directed edge to every original vertex. This is enough to turn both classical diameter and min-diameter infinite, while leaving the reachability diameter completely unchanged. Furthermore, computing shortest paths from the newly added vertices gives us no information about the original graph; the same holds for balls of fixed size or distance, as commonly used in diameter approximation algorithms. Thus, any constant-factor approximation scheme that relies on the idea of iteratively sampling a vertex and then computing distances from this vertex would need to contain a mechanism to ignore these "distraction" vertices. To the best of our knowledge, the existing diameter toolkit cannot handle and distinguish this case.

The more involved algorithms for classical diameter (e.g., [36]) and MinDiam (e.g., [3] and [15]) will still resist adaptation to ReachDiam for similar reasons. This is because, at their core, they still require computing shortest paths from and to a small number of carefully chosen vertices (potentially in a recursive subinstance or in limited size balls) to be informative about the diameter.

We will see that with techniques based on sampling vertices or hitting sets, we *can* obtain polynomial additive approximations for reachability diameter in unweighted graphs in $o(mn)$ time, where there is a trade-off between the computation time and the approximation. However for the reasons described, it seems quite challenging to get rid of these larger additive factors for general graphs. This implies that we get better approximations only when the reachability diameter is known to be large, but getting good estimates when the diameter is small remains a challenge. We are, however, able to avoid these bad instances and get good approximations for special graph classes by using chain-cover based or tree decomposition based approaches.

Connections with Shortcut sets and Hopsets. As discussed earlier, we noted that the reachability diameter is the appropriate notion of diameter when we consider objects such as shortcut sets and hopsets. In fact, if after computing a shortcut set (or hopset), we would like to verify that we have correctly reduced the diameter to the desired parameter, we would need to estimate the reachability diameter.

Interestingly, while we cannot show any blackbox reductions between computing shortcut sets and reachability diameter, we do establish some technical connections. We see that using the same ideas as in shortcut sets of [34] gives us algorithms where the approximation factor for reachability diameter correspond to the hopbound achieved in the same time.

Building on this connection, we give a new exact hopset construction and a simplified algorithm for shortcut sets for graphs with bounded treewidth.

2 Our Results

Our goal is to characterize the fine-grained complexity landscape of (approximate) reachability diameter computation both from a lower bound and upper bound point of view. For some settings, these results are tight (such as for approximating ReachDiam in weighted graphs), for

Table 1 Overview of approximation algorithms for reachability diameter on DAGs. Det. abbreviates deterministic. The approximation is given as a multiplicative factor except for a tuple (a, b) which indicates a a-multiplicative and b-additive approximation.

Weighted?	Approx.	Det.?	Running time	Ref.	Technique
✓	exact	✓	$\tilde{\mathcal{O}}(nm)$		$n \cdot$ BFS
✓	exact	✓	$\tilde{\mathcal{O}}(m \cdot \mathrm{tw} + m^{1+o(1)})$	Thm. 16	Separator D&C like [3]
✓	$1+\varepsilon$	✓	$\tilde{\mathcal{O}}(n^\omega/\varepsilon)$		Round + APSP-approx
✓	3	✓	$\tilde{\mathcal{O}}(m \cdot \lvert \mathrm{MCC} \rvert)$	Cor. 3	Path-based approach
x	exact	✓	$\tilde{\mathcal{O}}(n^\omega)$		Exact transitive closure
x	$(1, k)$	x	$\tilde{\mathcal{O}}(nm/k)$	Thm. 7	Random hitting set
x	$(3, k)$	✓	$\tilde{\mathcal{O}}(nm/k)$	Thm. 8	ℓ-cover + path-based
x	$n^{1/2+o(1)}$	x	$\tilde{\mathcal{O}}(m)$	Thm. 9	Sampling pivots

others gaps remain (such as for unweighted graphs). While the main upper bound techniques do not translate to our setting for constant-approximation of ReachDiam due to described technical challenges, we do show several upper bounds either with larger approximation for general graphs or with small approximation for special graph classes.

We outline the main results here; see Table 1 for a summary of our upper bounds and Table 2 for our lower bounds. A main distinction for both upper and lower bounds will be if they apply to weighted or unweighted directed graphs. When we discuss optimality, we always do so up to subpolynomial factors.

2.1 Weighted Graphs

For weighted graphs, the worst-case complexity landscape is relatively simple. Let us start with the main hardness result (see Section 3.1 for the definitions of the hardness assumptions).

▶ **Theorem 1.** *Unless the simplicial vertex conjecture fails, there is no poly(n)-approximation for* ReachDiam *on weighted directed graphs (even on DAGs) running in time $\mathcal{O}(n^{\omega-\varepsilon})$.*

The result holds even if the approximation guarantee is a (at most singly exponential) function of the input graph size. That is, we also rule out even factor-n approximations.

This means that the optimal combinatorial algorithm is running Dijkstra's algorithm from all vertices (which will calculate ReachDiam exactly). The standard algebraic $(1+\varepsilon)$ All-Pairs-Shortest-Paths approximation [41] can be used to approximate ReachDiam. As we only care about the maximum finite distance, we can get rid of the dependency on the largest edge weight using a rounding technique, getting a $\tilde{\mathcal{O}}(n^\omega/\varepsilon)$ running time bound. This was previously observed for standard diameter in [12]. We show the same hardness bound based on a different hardness assumption, namely, the high-dimensional orthogonal vectors problem. Hence, we put the hardness result on a more secure footing in case one of these conjectures turn out to be false.

▶ **Theorem 2.** *Suppose $\omega > 2$ and consider any $\epsilon > 0$. Unless high-dimensional OV fails, there is no algorithm with poly(n)-approximation for* ReachDiam *on weighted graphs running in time $\mathcal{O}(n^{\omega-\varepsilon})$.*

Hence, under these fine-grained assumptions, the straightforward approximate APSP upper bound matches the lower bound on worst-case instances. It is therefore natural to see if the problem remains hard in certain graph classes. In particular, we take a parameterized

view at the fine-grained complexity of the problem. This leads us to near-linear constant-factor approximation algorithms given one of these parameters is bounded. For these graph classes, we thus side-step the general impossibility of constant-factor approximations in less than matrix-multiplication time.

First, for DAGs, we show that if the *width* (the size of the minimum number of chains that cover all vertices) is bounded, we can compute a 3-approximation in almost-linear time.

▶ **Corollary 3.** *There is an algorithm that, given a DAG G, computes a 3-approximation for the* ReachDiam *in time $\tilde{\mathcal{O}}(|\text{MCC}|\, m + m^{1+o(1)})$, where $|\text{MCC}|$ is the width of G.*

The width is a standard parameter for studying problems related to reachability in DAGs. For example, they are also widely considered for studying shortcut sets [29, 31]. The algorithm underlying our result is inspired by the chain-cover-based approaches widely used in recent advancements in that area. Crucially, for a chain C, we can approximate the diameter among all demand pairs that have a shortest path passing through C in $\tilde{\mathcal{O}}(m)$ time.

As a second parameter, we look at the treewidth of the underlying undirected graph of the input directed graph. This parameter was originally considered for diameter problems by Abboud, Vassilevska Williams, and Wang [3] in the context of fixed parameter subquadratic algorithms. We share their motivation of exploring structural parameters with the intent of overcoming barriers from fine-grained complexity. In fact, we can lightly adapt one of their algorithms for the classic diameter definition to compute the ReachDiam exactly in almost-linear time for graphs of bounded treewidth. We first give a much simpler version of this algorithm that already yields a 2-approximation of ReachDiam.

Exact hopsets. We then adapt this algorithm to compute shortcut sets and *exact* hopsets that have constant hopbound and linear size for graphs of bounded treewidth. At its core, this result relies on the fact that any graph of bounded treewidth will recursively have small balanced separators. This yields a recursive technique where the number of shortcut edges relates to the size of these balanced separators. Formally, we show the following result.

▶ **Theorem⋆ 4.** *There is an algorithm that given a directed graph G of treewidth* tw*, computes an exact hopset of hopbound* 2 *of size $\tilde{\mathcal{O}}(n \cdot \text{tw})$ in time $\tilde{\mathcal{O}}(m \cdot \text{tw} + m^{1+o(1)})$.*

Compare this to the general bounds: a lower bound due to Bodwin and Hoppenworth shows that there are graphs where we cannot construct a $\mathcal{O}(n)$-size shortcut set with stretch better than $n^{1/4}$ [10]. For exact hopsets, they even show that there are graphs where there cannot be a $\mathcal{O}(n)$-size set with hopbound better than $\sqrt{n}$.

By the folklore connection between TC-spanners and shortcut sets, a result like Thm. 4 was already implied by a more complicated algorithm for H-minor-free graphs by [9]. Hence, we get a simpler algorithm for shortcut sets. For exact hopsets, this is the first such result.

Recently, Chalermsook, Jiang, Mukhopadhyay, and Nanongkai [14] raised the question of finding efficient algorithms computing better shortcut sets and TC-spanners in restricted graph classes, and explicitly suggested low-treewidth graphs as such a class to explore. We observe that the result in [BGJ+] implicitly gives such improved bounds for bounded treewidth graphs, which is also implied by our simplified approach. Moreover, we are addressing this question for hopsets. Both for shortcut sets and exact hopsets, these provide small constant stretch in the most-commonly studied near-linear size regime.

Table 2 Overview of our lower bounds.

Weighted?	Approx.	Time lower bound	Density	Hypothesis	Ref.
✓	poly(n)	$n^{\omega-o(1)}$	Dense	SV	Thm. 1
✓	poly(n)	$n^{\omega-o(1)}$	Dense	HD-OV	Thm. 2
x	< 2	$n^{3-o(1)}$	Dense	BMM	Cor. 5
x	$< 3/2$	$m^{2-o(1)}$	Sparse	OV	Thm. 6

2.2 Unweighted Graphs

In unweighted graphs, based on standard hardness assumptions, we cannot obtain small approximation factors more efficiently than the $(1+\varepsilon)$ approximation based on matrix multiplication.

▶ **Corollary 5.** *Unless combinatorial BMM fails, there is no combinatorial better-than-2 approximation algorithm for* ReachDiam *running in time* $\mathcal{O}(n^{3-\varepsilon})$*, even on unweighted DAGs.*

Similarly unless BMM fails, there is no better-than-2-approximation for ReachDiam *running in time* $\mathcal{O}(n^{\omega-\varepsilon})$*.*

This can be compared to the simple exact algorithm for unweighted ReachDiam in time $\tilde{\mathcal{O}}(n^\omega)$ that binary searches the smallest power of the adjacency matrix that forms the transitive closure.

Interestingly, we can prove a lower bound for a different time/approximation tradeoff, based on the orthogonal vectors conjecture, which implies SETH. In particular, we show that the ReachDiam approximation remains hard even in sparse unweighted DAGs.

▶ **Theorem 6.** *Unless OV fails, there is no better-than-3/2 approximation algorithm for* ReachDiam *running in time* $\mathcal{O}(m^{2-\varepsilon})$*, even on unweighted DAGs.*

Hence, our lower bounds for unweighted graphs are not as strong as the lower bounds for weighted graphs (w.r.t approximation factor). In Section 2.3 we explain why it is technically challenging to obtain a stronger lower bound for unweighted graphs.

On the other hand, we can directly exclude certain lower bounds by giving the following upper bounds. We use a standard sampling argument to get the following randomized result.

▶ **Theorem 7.** *There is an algorithm that given a parameter* $k \in \mathbb{N}$*, outputs an estimate* $\hat{D}$ *such that with high probability,* $\text{ReachDiam}(G) - k \leq \hat{D} \leq \text{ReachDiam}(G)$*. It always runs in time* $\mathcal{O}(nm/k)$*.*

The additive term in the approximation will become polynomial if we want faster algorithms, as discussed. It seems challenging to avoid such terms with standard techniques.

Using ideas from the chain-cover-based approach used in the shortcut set literature mentioned above, we can derandomize this algorithm. The core subroutine is similar to our MCC-based result, but while |MCC| is a property of the input graph, here we use a chain-cover where we can control its size. A smaller chain-cover must then tolerate a larger approximation error, giving the stated trade-off.

▶ **Theorem 8.** *There is a deterministic algorithm that given a DAG* G *and a parameter* ℓ*, outputs an estimate* $\hat{D}$*, such that* $(\text{ReachDiam}(G) - \ell)/3 \leq \hat{D} \leq \text{ReachDiam}(G)$ *and runs in time* $\tilde{\mathcal{O}}(nm/\ell + m^{1+o(1)})$*.*

Seeing these upper bound results, it is natural to ask whether we can improve our lower bound results to show that the trade-off between a k-approximation and $\mathcal{O}(nm/k)$ time is the best that one can hope for (at least combinatorially). The following result, obtained by modifying a shortcut set algorithm due to Jambulapati, Liu, and Bernstein [34], shows such a lower bound trade-off does not hold. This also establishes a technical connection between shortcut set computation and reachability diameter.

▶ **Theorem 9.** *There is an algorithm computing a $n^{1/2+o(1)}$-approximation of* ReachDiam *in $\tilde{\mathcal{O}}(m)$ time.*

At its core, the algorithm relies on sampling pivot vertices and then building recursive subinstances based on the reachability relationship between each vertex and all pivots. This recursion is lossy in the sense that the diameter might be split up into different subinstances (thus the maximum diameter of all subinstances might be less than that of the input graph). The quality analysis now relies on tuning the recursion to have this bad event happen the fewest times while maintaining near-linear running time.

2.3 Open Problems and Technical Challenges

We are left with two important open problems about the reachability diameter. The most natural one is on bridging the upper bound and lower bound gap for unweighted graphs, which we formally state below. Secondly, we discuss whether this problem can be generally reduced to DAGs.

Best Approximation for Unweighted Graphs. Our strong lower bounds for the weighted setting indicate that it is unlikely to find fast approximation algorithms there. In particular, by computing $(1+\epsilon)$-APSP (all pairs shortest paths) in time $\tilde{\mathcal{O}}(n^\omega/\varepsilon)$ we can essentially match the lower bounds in Theorems 1 and 2.

However, in the unweighted setting, the picture is not complete. The ReachDiam of a graph can be computed exactly with fast matrix multiplication and this cannot be improved polynomially due to the reduction to BMM. Furthermore, even better-than-2 approximations are ruled out in faster time. On the other hand, we show that faster algorithms with polynomial approximations are possible (Theorem 7 and Theorem 9). This motivates the following open question.

▶ Open Problem 1. What is the best approximation/running time tradeoff for ReachDiam in unweighted graphs?

In particular, what is the best approximation possible in either linear-time or in better than $O(n^\omega)$ time?

It is also possible to answer Open Problem 1 by proving a stronger lower bound, such as the ones shown in Theorems 1 and 2. However, such a lower bound requires new techniques, as we run into a challenge that was previously called the "triangle-inequality barrier" [1, 27, 37].

Concretely, to prove a fine-grained lower bound for a c-approximation for ReachDiam we need to construct a graph for which the YES and NO instances have the following property: In one case they contain short paths of length at most ℓ for every reachable pair of vertices. In the other case, there should be a reachable pair of vertices u, v with distance $c \cdot \ell$.

In a straight-forward way we could create a path of length $c\ell$ from u to v but shorter paths would exist in the first case to ensure a smaller distance. However, there must be intermediate vertices on this path as the graph is unweighted. They would introduce distances up to $c\ell - 1$, making this simple approach implausible. Overcoming this barrier to prove stronger hardness results could therefore reveal interesting techniques how to achieve the same for other distance problems.

Reducing General Directed Graphs to DAGs. Conceptually, the hardest instance for computing reachability diameter seems to be when the input is a DAG. One explanation is that our lower bound instances are all DAGs. Another intuition is that we can consider the DAG of SCCs, and run the algorithms for classical diameter on each component. However this does not directly lead to a reduction without losing substantially in the approximation. Our next open problem aims to formalize this intuition that such a reduction exists.

▶ Open Problem 2. Can we reduce ReachDiam in general directed graphs to ReachDiam in DAGs?

Such a reduction would immediately allow us to transfer our results based on chain-covers in Corollary 3 and Theorem 8 to general graphs. It would also allow us to restrict ourselves to DAGs in Open Problem 1. Furthermore, all our hardness instances are DAGs, so for example Theorems 1 and 2 imply that DAGs are as hard as general directed graphs for weighted approximation.

Recently Assadi, Hoppenworth, and Wein [5] proved a similar result for distances in directed graphs. They construct $\mathcal{O}(\log n)$ DAGs with $\tilde{\mathcal{O}}(m)$ additional total edges (see also [23]). Then, they guarantee that distance $d(u, v)$ in the original graph is poly-logarithmically approximated by the minimum distance from u to v in the collection of DAGs. Hence, for problems like directed approximate shortest paths with decently large approximation factor, we can assume without loss of generality that the given input graph is a DAG.

However, this result does not solve Open Problem 2, since any individual DAG could have a much larger reachability diameter than the original graph. Therefore, we need a slightly different guarantee, and it would be interesting if it is possible to transfer the concept of DAG covers to reachability diameter approximation.

The proofs of statements marked $\star$ are omitted and can be found in the full version, linked on the title page.

3 Preliminaries

For $n \in \mathbb{N}$, write $[n] \coloneqq \{1, \ldots, n\}$. For a vertex v in some graph, we denote its neighborhood as $N(v)$. In directed graphs, we write its out- and in-neighborhoods as $N^+(v)$ and $N^-(v)$, respectively. Note that then $N(v) = N^+(v) \cup N^-(v)$. The *transitive closure* of a graph $\mathrm{TC}(G)$ is defined as the graph $(V(G), \{(u, v) \in V(G)^2 \mid u \text{ can reach } v \text{ in } G\})$. A *chain* is a path in the transitive closure of a graph. In a directed graph G, a *chain* is a sequence of vertices $v_1, \ldots, v_\ell$ such that for all $i \in [n-1]$, we have $v_i \to v_{i+1} \in \mathrm{TC}(G)$. Each DAG G encodes a poset $\leq_G$, that is, for vertices u, v we write $u \leq_G v$ if there is a path from u to v in G. Where it is clear from context, we omit the subscript in $\leq_G$.

The *width* of a DAG is the size of its minimum chain cover, that is the minimum number of chains required to cover every vertex. It is a common graph parameter used to study the complexity of problems on DAGs [29, 31, 13, 30, 35]. Equivalently it can be defined as the size of the largest independent set in the transitive closure (these are also known as *anti-chains*). The *treewidth* of an undirected graph is the minimum largest bag size of any tree-decomposition minus one. A tree decomposition of a graph G is a tree T where every vertex $X \in V(T)$ is associated with a subset of the vertices in G. We call these sets *bags* and treat the tree-vertices directly as their bags. The following three properties hold: (1) for every $v \in V(G)$, there is a bag $X \in V(T)$ s.t. $v \in X$, (2) for every edge $\{u, v\} \in E(G)$, there is a bag $X \in V(T)$ s.t. $u, v \in X$, and (3) for all $v \in V(G)$, the bags of T that include v form a (connected) subtree of T. Graph families with bounded treewidth include cacti, series-parallel graphs, and outerplanar graphs. For a more complete discussion of treewidth,

see [16]. While there are many attempted definitions of the treewidth of a directed graph, we will only use it to mean the treewidth of the underlying undirected graph (i.e., the graph obtained by forgetting all edge directions). A *c-balanced separator* in a graph G is a set S such that each connected component in $G - S$ has at most $c|V(G)|$ vertices. We omit c if it is some constant in $(0, 1)$ that is independent of the graph and its size.

3.1 Hypotheses for Conditional Lower Bounds

We obtain our lower bound results by fine-grained reductions from more or less standard hypotheses from fine-grained complexity. We dedicate this subsection to their definition as well as some context for each hypothesis

Boolean Matrix Multiplication. The combinatorial boolean matrix multiplication (BMM) hypothesis reads as follows.

▶ Hypothesis 1 (Combinatorial BMM). Given vectors $A, B \in \{0,1\}^{n\times n}$, there is no combinatorial[1] algorithm that computes $C \in \{0,1\}^{n\times n}$ with $C_{ij} = \bigvee_k A_{ik} \wedge B_{kj}$ in time $\mathcal{O}(n^{3-\varepsilon})$, for any $\varepsilon > 0$.

Algebraically, that is, using fast matrix multiplication, combinatorial BMM can clearly be solved in time $\mathcal{O}(n^\omega)$ by a single matrix multiplication for $\omega < 2.372$ [4]. This is also the best general algorithm for BMM and it is generally assumed that no better algorithm exists [7, 17]. This is why we also use BMM to prove conditional n^ω lower bounds. Restricting our attention to combinatorial algorithms, a truly subcubic algorithm for triangle detection in tripartite graphs would imply the same for BMM [38]. Hence, we use that problem for hardness results under Hypothesis 1.

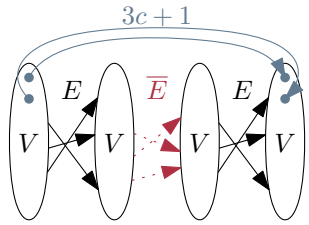
Orthogonal Vectors. The orthogonal vectors (OV) hypothesis is one of the most important hypotheses in fine-grained complexity as is was used to prove many lower bounds (see [40] for a summary) and follows from the strong exponential time hypothesis [39]. Most importantly for our case, it is a well-suited hypothesis for diameter lower bounds and has been used in different variants for standard diameter [36, 6, 11, 33] and MinDiam [3]. Our OV-based hardness result follows these previous hardness constructions.

▶ Hypothesis 2 (OV). Given sets A, B of size n of vectors in $\{0,1\}^d$, with $d \geq \omega(\log n)$ there is no algorithm that decides if there is $a \in A$ and $b \in B$ with $\sum_i a_i b_i = 0$ in time $\mathcal{O}(n^{2-\varepsilon})$, for any $\varepsilon > 0$.

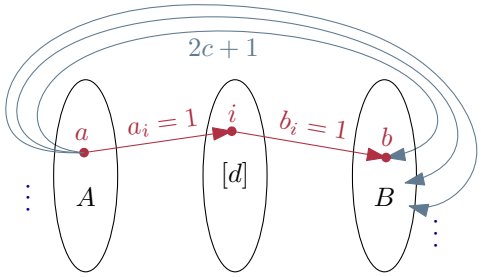
High-Dimensional OV. The high-dimensional OV hypothesis (HD-OV) considers the OV problem for higher dimensions, that is, we now have $d = \Theta(n)$. It can be thought of as an analogue of the OV hypothesis for dense graphs. In this setting, fast matrix multiplication produces a running time of $\mathcal{O}(n^\omega)$, a speedup over the brute-force-way to solve standard OV.

This problem was previously used as the basis of a lower bound for MinDiam by Dalirrooyfard and Kaufmann [18]. They showed that a better-than-3/2-approximation for MinDiam requires time $\Omega(n^{\omega-o(1)})$.

[1] "Combinatorial" algorithms generally informally describe algorithms that avoid the use of fast matrix multiplication techniques. See for example [2] for a detailed discussion of the term and motivations behind combinatorial algorithms and lower bounds.



(a) Lower bound construction from Theorem 1. The graph has ReachDiam 3 iff no vertex is simplicial, that is, the neighborhood of no vertex is a clique.



(b) Lower bound construction from Theorem 2. The graph has ReachDiam 2 iff there is no orthogonal pair.

Figure 1 Our lower bound constructions for weighted ReachDiam.

▶ Hypothesis 3 (HD-OV). Given sets A, B of size n of vectors in $\{0,1\}^d$ with $d = \Theta(n)$, there is no algorithm that decides if there is $a \in A$ and $b \in B$ with $\sum_i a_i b_i = 0$ in time $\mathcal{O}(n^{\omega-\varepsilon})$, for any $\varepsilon > 0$.

Simplicial Vertex. The simplicial vertex problem (SV) is a graph problem that asks you to decide if there is a vertex whose neighborhood is a clique. It can again be solved in fast matrix multiplication time [28]. The problem has been used as a basis for lower bounds for the clique cutset problem in [32], for lower bounds related to n-pairs diameter [17], and for undirected diameter in structured graphs [21].

▶ Hypothesis 4 (SV). Given an undirected, unweighted graph G, there is no algorithm that decides if there exists a vertex $v \in V(G)$ such that $N(v)$ is a clique in time $\mathcal{O}(n^{\omega-\varepsilon})$, for any $\varepsilon > 0$.

The usefulness of this hypothesis stems from two main properties. First is its quantifier structure of $\exists\forall$ or $\forall\exists$ when negated, which is the same structure that is necessary for upper bounding the diameter (for all vertex pairs, there is a short path). Secondly, the first quantifier quantifies over a set of size n which is necessary to avoid a blow-up in the reduction in our case.

4 Weighted Graphs

In this section we prove our results for weighted graphs. We start with strong lower bound for any approximation algorithm for ReachDiam in Section 4.1. Our positive results are hence restricted to special graph classes. In Section 4.2, we give an algorithm for DAGs of small width. In Section 4.3, we go on to give an algorithm for small-treewidth graphs and highlight connections to shortcut sets and hopsets.

4.1 Lower Bounds

First, we prove our lower bound based on the simplicial vertex hypothesis. We use the hypothesis in a similar way to [17, 21]. See Section 3.1 for a definition and discussion.

▶ **Theorem 1.** *Unless the simplicial vertex conjecture fails, there is no poly(n)-approximation for* ReachDiam *on weighted directed graphs (even on DAGs) running in time $\mathcal{O}(n^{\omega-\varepsilon})$.*

Proof. We show that such an approximation algorithm would imply the simplicial vertex hypothesis is false. Let $c = \text{poly}(n)$ be the approximation factor of that algorithm and let G be an undirected input graph we are to determine the existence of a simplicial vertex in.

As input to our ReachDiam approximation algorithm, create a graph H as follows.

- Let V_1, V_2, V_3, V_4 be four disjoint copies of $V(G)$ and set $V(H) \coloneqq V_1 \sqcup \cdots \sqcup V_4$.
- For any edge $\{u, v\} \in E(G)$, create edges $v_1 \to u_2$, $u_1 \to v_2$, $v_3 \to u_4$, and $u_3 \to v_4$, where the indices indicate the copies in the sets V_1 to V_4.
- For any non-edge $\{u, v\} \notin E(G)$, create edges $v_2 \to u_3$, and $u_2 \to v_3$. All these edges have weight 1.
- Additionally, for every $v \in V(G)$, create an edge $v_1 \to v_4$ with weight $3c + 1$.

We prove that this graph has a ReachDiam of 3 if there is no simplicial vertex and $3c + 1$ otherwise. Therefore a $\mathcal{O}(n^{\omega-\varepsilon})$ c-approximation yields a $\mathcal{O}(n^{\omega-\varepsilon})$ algorithm to decide the existence of a simplicial vertex.

First, observe that for any pair of vertices in H that are not the V_1 and V_4 copy of the same vertex in G, either no connection exists or the connection only uses at most three edges of weight 1. Thus, these pairs are either not reachable or have distance at most 3. Therefore, we can ignore them for the rest of the analysis and focus on the distance between the V_1 and V_4 copies of the same vertex from $V(G)$.

Now, assume that G has a simplicial vertex v. Our goal is to show that the path consisting of the single edge $v_1 \to v_4$ is a shortest path with length $3c + 1$ and thus $\text{ReachDiam}(H) \geq 3c + 1$. Assume for the sake of contradiction, that there is a shorter path from v_1 to v_4. Then, since H is a layered graph with 4 layers (except the direct edges between V_1 and V_4), this path must be of the form $v_1 \to a_2 \to b_3 \to v_4$, where $a_2 \in V_2$ and $b_3 \in V_3$. By construction of H, the edges $v_1 \to a_2$ and $b_3 \to v_4$ mean that $a, b \in N_G(v)$. The edge $a_2 \to b_3$ means that $\{a, b\}$ is *not* an edge in G. Therefore, v is not a simplicial vertex in G, contradicting our assumption and we can conclude that $d_H(v_1, v_4) = 3c + 1$ and $\text{ReachDiam}(H) \geq 3c + 1$, as desired.

Now, assume that G has no simplicial vertex. Let v be an arbitrary vertex. Because v is not a simplicial vertex, there are vertices $a, b \in N(v)$ such that $\{a, b\}$ is not an edge in G. Therefore $v_1 \to a_2 \to b_3 \to v_4$ is a path of length 3 in H and thus $d_H(v_1, v_4) \leq 3$. Since we chose v arbitrarily, we have that $\text{ReachDiam}(H) \leq 3$. ◀

Our next lower bound proves the same bound but based on the high-dimensional OV hypothesis instead. It hence strengthens the previous theorem.

▶ **Theorem 2.** *Suppose $\omega > 2$ and consider any $\epsilon > 0$. Unless high-dimensional OV fails, there is no algorithm with poly(n)-approximation for* ReachDiam *on weighted graphs running in time $\mathcal{O}(n^{\omega-\varepsilon})$.*

Proof. We use the standard OV graph construction to reduce high-dimensional OV to ReachDiam approximation. Assume there is an algorithm with approximation factor $c \in \mathbb{R}^+$ for ReachDiam in time $\mathcal{O}(n^{\omega-\varepsilon})$ for some $\varepsilon \in (0, \omega - 2]$. See Figure 1b for a drawing of the reduction.

Let $A, B \subseteq \{0, 1\}^d$, with $|A|, |B| = n$ and $d \in \Theta(n)$. Create a graph as follows.

- Create $V \coloneqq A \sqcup B \sqcup [d]$.
- For $a \in A$ and $i \in [d]$, create an edge $a \to i$ iff $a_i = 1$.
- Similarly, for $b \in B$, create an edge $i \to b$ iff $b_i = 1$.
- All of these edges have weight 1.
- Additionally, for all $a \in A, b \in B$ add an edge $a \to b$ of weight $2c + 1$.

Note that between any two a, b there now is a path of length 2 via $[d]$ if and only if a and b are *not* orthogonal. Because of the last edges, an orthogonal pair of vectors has distance $2c + 1$. On the other hand, ReachDiam > 2 also implies that there is an orthogonal pair of vectors.

Since the graph has $\mathcal{O}(n)$ vertices, $\mathcal{O}(n^2)$ edges, and $\varepsilon \leq \omega - 2$, the running time of the approximation algorithm dominates. Hence, a c-approximation in time $\mathcal{O}(n^{\omega-\varepsilon})$ can also decide high-dimensional OV in the same time. ◀

4.2 Chain-Based Algorithm

The width of a DAG G gives a decomposition of G into chains. In the introduction, we discussed how the ReachDiam of a DAG is equivalent with its MinDiam if the DAG consists of a single path plus extra edges along that path. This is equivalent to having width one. Thus, we can use algorithms for MinDiam to compute a constant-factor approximation of ReachDiam in subcubic time for DAGs of width one. This raises the natural question of whether we can extend this idea to larger widths. Recall that the width is the size of the minimum chain cover. The rough strategy is thus as follows:

1. Decompose G into chains.
2. Use MinDiam-inspired techniques to approximate ReachDiam between vertices on the same chain (will be Lemma 11).
3. Use additional ideas to approximate ReachDiam between vertex pairs on different chains (will be Lemma 12).
4. Combine this information (will be Theorem 13).

In this section, we will see how to implement this strategy. We can achieve step (1) using a result by Cáceres.

▶ **Theorem 10** ([13]). *There is an $\mathcal{O}(m^{1+o(1)})$ time algorithm that computes a minimum chain cover of a DAG.*

So, let us now look at step (2) of our strategy. The following lemma extends Theorem 2.2 from [3]. We show that with a slight extension their algorithm can be used on any chain inside a DAG to approximate the maximum distance between any two vertices on the chain. Note that the shortest path between (even sequential) vertices on the chain might use intermediate vertices not on the chain, thus requiring us to go slightly beyond the original algorithm by [3].

▶ **Lemma 11.** *Given a DAG G and a chain $C = v_1, \ldots, v_\ell$, Algorithm 1 computes a 2-approximation of the maximum distance of any two sequential vertices on the chain in $\tilde{\mathcal{O}}(m)$ time.*

Proof. For the running time analysis, observe that this divide and conquer algorithm generates two subproblems, each with at most half as many vertices. In each recursive call, the additional overhead (a topological sort, a forwards and backwards Dijkstra) runs in $\mathcal{O}(n + m)$. Solving this recurrence leaves us with a total running time of $\mathcal{O}(m \log n)$.

To prove correctness, assume that a and b are two vertices on the chain with maximum pair-wise distance. Assume wlog. that a appears before b on the chain (and thus in the topological order). In each recursion step, v_k can be before both a and b, after both, or between them. It the first two cases, a and b are in the same recursive subproblem, so let us focus on the case where $a \leq v_k \leq b$ for the first time. Let V' be the vertex set of this recursive call. Since $a, b \in V'$, note that V' must contain the entire subchain from a to b.

Algorithm 1 2-approximation of intra-chain diameter.

Input: DAG G, chain $C = v_1, \dots, v_\ell$, numbered in topological order
Output: 2-approximation for the maximum distance of two vertices in C

1. Compute a topological order O.
2. Remove all vertices before v_1 or after v_ℓ in O.
3. Let V_1, V_2 be a partition of $V(G)$ such that $V_1 = \lfloor |V| / 2 \rfloor$ and no vertex from V_1 appears after any vertex from V_2 in O. Let v_k be the last vertex from C in V_1.
4. Run Dijkstra's algorithm forwards and backwards from v_k.
5. Run the algorithm recursively on $G[V_1]$ with chain $v_1, \dots, v_k$ and $G[V_2]$ with chain $v_{k+1}, \dots, v_\ell$.

Report the maximum distance of any vertex from the chain in any shortest path tree seen during the execution.

Similarly, since the divide-steps respect the topological order, any vertex that is on a path between a and b must be in V'. Thus $d_G(a,b) = d_{G[V']}(a,b)$. By, the triangle inequality, we have $d(a,b) \le d(a,v_k) + d(v_k,b)$. By this and the choice of a and b, we can conclude that

$$\max(d(a,v_k), d(v_k,b)) \le d(a,b) = \max_{u \le v \in C} d(u,v) \le 2\max(d(a,v_k), d(v_k,b)).$$

By the definition of the output of the algorithm, this shows the desired property.

Clearly, any distance that the algorithm outputs corresponds to the length of a path between two vertices on the chain. Therefore, the algorithm can never overestimate the diameter. ◀

Next, we show how to compute an estimate for ReachDiam with the starting vertex on the chain. This is step (3) of our plan.

▶ **Lemma 12.** *There is an algorithm that given a directed graph G and a chain $C = v_1, \dots, v_\ell$, computes an estimate $\hat{D}$ such that $\hat{D} \le \max_{a \in C, b \in V(G)} d(a,b) \le \hat{D} + \max_{a,b \in C} d(a,b)$ in $\tilde{\mathcal{O}}(m)$ time.*

Proof. We start by computing the shortest path trees $T_i \coloneqq \textsc{Dijkstra}(v_i, G \setminus \bigcup_{j=i+1}^{\ell} T_i)$ with $T_\ell = \textsc{Dijkstra}(v_\ell, G)$. By working backwards, starting from $i = \ell$, this is possible in time $\tilde{\mathcal{O}}(m)$. We simply keep track of the current tree and already visited vertices by other trees. So each edge will be visited at most once. Then, we report the height of the highest tree as the estimate $\hat{D}$.

We prove the two inequalities on $\hat{D}$ separately and start with $\hat{D} \le \max_{a \in C, b \in V(G)} d(u,v)$. Towards this goal, assume T_i is one of the highest trees, and b is a vertex in its last layer. Recall that v_i is the root of T_i. To show the inequality, we show that the path from v_i to b in T_i is a shortest path. Assume by contradiction that it is not. Then the other path that is shorter must contain a vertex u that is in a T_j with $j > i$ (because otherwise u would be in T_i). But then since u can reach b and v_j can reach u, we also have that v_j can reach b and therefore b must be in one of the trees $T_j, \dots, T_\ell$, contradicting our assumption.

Let us now prove that $\max_{a \in C, b \in V(G)} d(a,b) \le \hat{D} + c$, with $c = \max_{a,b \in C} d(a,b)$. Let $a \in C, b \in V(G)$ be one of the pairs maximizing $d(a,b)$. Assume $a = v_i$ and let T_j be the tree such that $b \in T_j$. Then $a \rightsquigarrow v_j \rightsquigarrow b$ is a path of length at most $c + \hat{D}$ and therefore $d(a,b) \le \hat{D} + c$, as desired. ◀

Now, we combine the two previous lemmas. We compute estimates for both the diameter on the chain as well as the diameter leaving the chain. These can then be combined to give a 3-approximation for any path with one endpoint on the chain. This is step (4).

▶ **Theorem 13.** *There is an algorithm that given a DAG G and a chain C in G computes an estimate $\hat{D}$ with $\hat{D} \leq \max_{a\in C, b\in V(G)} d(a,b) \leq 3\hat{D}$ in time $\tilde{\mathcal{O}}(m)$.*

Proof. We run the algorithm from Lemma 12 to get an estimate $\hat{D}$, and the algorithm from Lemma 11. From the set of shortest path trees, we take the maximum height h of any tree and return the maximum $\hat{D}^* := \max\{h, \hat{D}\}$.

The runtime is clearly as claimed. By Lemma 11, $\hat{D}$ is always bounded by some distance on the chain. Furthermore, h is a valid distance, since anything reached in a shortest path tree is not in a shortest path tree from a later vertex on the chain. Hence, the first inequality holds. On the other hand, we have

$$\max_{a\in C, b\in V(G)} d(a,b) \leq \max_{a,b\in C} d(a,b) + h \leq \hat{D}^* + 2\hat{D}^* = 3\hat{D}^*. \qquad \blacktriangleleft$$

Computing a minimum chain cover using Theorem 10 and then applying Theorem 13 to every chain in it gives the following result.

▶ **Corollary 3.** *There is an algorithm that, given a DAG G, computes a 3-approximation for the* ReachDiam *in time $\tilde{\mathcal{O}}(|\mathrm{MCC}|\, m + m^{1+o(1)})$, where $|\mathrm{MCC}|$ is the width of G.*

4.3 Treewidth-Based Algorithm

In this section, we take a structural perspective and explore the relationship between diameter estimation and treewidth. We also deepen the connection between shortcut sets, hopsets and diameter estimation under this lens. First, we briefly explain how an algorithm from [3] can be slightly modified to exactly compute the ReachDiam of a directed graph whose underlying undirected graph has bounded treewidth. We then give a simplified version of that algorithm that yields as 2-approximation and has a polynomial dependence on the treewidth (compared to the exponential dependence in [3]). Their approach relies on a complicated data structure that we replace with a simple, explicit construction to obtain a shortcut sets. We then show how this leads to exact hopsets with stretch 2 of size $\tilde{\mathcal{O}}(n \cdot f(\mathrm{tw}))$.

The core fact exploited by all these techniques is that an undirected graph of bounded treewidth has a small bounded separator. Since this holds recursively, we can split the graph in a balanced way, while handling all paths that touch the bounded-size separator. Interestingly, this separator of the underlying undirected graph is useful for computing the (reachability) diameter in directed graphs and for TC-spanners, shortcut sets and hopsets. Formally, we will use the following lemma.

▶ **Lemma 14** (Lemma 7.19 of [16])**.** *Let T be a tree decomposition of an undirected graph G of bag size k. Then there is a $1/2$-balanced separator of G of size k. It can be computed from T in linear time.*

Exact ReachDiam. Let us start by remarking how to adapt the algorithm in [3] to compute the reachability diameter. Unfortunately, we cannot state the result to be self-contained here without replicating much of their work, we thus settle for explaining the required modification.

▶ **Observation 15** (Cf. Theorem 3.3 of [3])**.** *There is an algorithm that given a directed graph G of treewidth* tw*, computes* ReachDiam(G) *in time $\tilde{\mathcal{O}}(m \cdot \mathrm{tw}^2 \log^{\mathrm{tw}-1}(n))$.*

Proof Sketch. As their algorithm computes the eccentricities of each vertex u in the graph as $\max_{v \in V(G)} d(u,v)$, we need to adapt it such that only finite distances are considered in this maximum. One can achieve this by only inserting finite values in to the range searching for maximum data structure they use internally. It is easy to verify this does not affect running time and the correctness stays essentially the same, except for the eccentricity definition. ◀

A Simpler 2-Approximation. Now let us take closer look at the algorithmic strategy of how to use treewidth by giving the simpler 2-approximation explicitly. Note that the linear dependence on treewidth is only reached by using recent fast max flow subroutines. Using different treewidth approximations from [20] a running time of $\tilde{\mathcal{O}}(m \cdot \mathrm{tw}^3)$ would also be achieveable, which is better for polylogarithmic treewidth.

▶ **Theorem⋆ 16.** *Algorithm 2 computes, given a directed graph G of treewidth* tw*, a 2-approximation of* ReachDiam(G) *in time* $\tilde{\mathcal{O}}(m \cdot \mathrm{tw} + m^{1+o(1)})$.

Algorithm 2 Approximation algorithm for the ReachDiam of a graph of bounded treewidth.

Input: Directed graph $G = (V, E)$
Output: 2-approximation of ReachDiam

1. Compute a tree decomposition T of G with bag size at most $\mathrm{tw} \cdot \log^3(n)$
2. Pick one of the bags S that is a balanced separator.
3. Run a forwards and backwards Dijkstra from every $s \in S$.
4. For every $s, s' \in S$, add an edge $s \to s'$ of weight $d(s, s')$ if that distance is finite.
5. For every component C of the undirected graph $G - S$, recurse on the directed graph $G[C \cup S]$.

Return the maximum distance seen (including the recursive instances).

Adaptation to Shortcut Sets and Hopsets. Let us now see how the previous algorithm can be adapted to yield shortcut sets and hopsets. Explicitly, we add the shortcut edges (or weighted hopset edges corresponding to distance of the endpoints) from and to the balanced separator in Step 3.

We focus our presentation on hopsets as the statement for hopsets directly implies the shortcut set version and as the shortcut set version, while it hasn't been explicitly stated, is implied by the result on TC-spanners in H-minor-free graphs in [9].

Recall, in a (non-negatively weighted) directed graph G, a (β, ε)-hopset is a set $H \subset V(G) \times V(G) \times \mathbb{R}$ of new (weighted) edges such that (1) no distance decreases from G to $G \cup H$ (including, becoming finite) and (2) in $G \cup H$ for all u, v with $d(u,v) < \infty$, there is a path in $G \cup H$ of β vertices of length $\leq (1+\varepsilon)d(u,v)$. If $\varepsilon = 0$, we have an exact β-hopset.

We mention that there is a previously studied special case of β-hopsets for $\beta = 2$, called *hub labelings*. In the language of hub labelings, similar connections as our Thm. 4 between distance problems and graph shortcuts have been made, see [22, 25].

▶ **Theorem⋆ 4.** *There is an algorithm that given a directed graph G of treewidth* tw*, computes an exact hopset of hopbound 2 of size $\tilde{\mathcal{O}}(n \cdot \mathrm{tw})$ in time $\tilde{\mathcal{O}}(m \cdot \mathrm{tw} + m^{1+o(1)})$.*

References

1 Amir Abboud, Mina Dalirrooyfard, Ray Li, and Virginia Vassilevska Williams. On Diameter Approximation in Directed Graphs. In *31st Annual European Symposium on Algorithms, ESA 2023*, pages 2:1–2:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.2.

2 Amir Abboud, Nick Fischer, Zander Kelley, Shachar Lovett, and Raghu Meka. New graph decompositions and combinatorial boolean matrix multiplication algorithms. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024*, pages 935–943. Association for Computing Machinery, 2024. doi:10.1145/3618260.3649696.

3 Amir Abboud, Virginia Vassilevska Williams, and Joshua Wang. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 377–391. SIAM, 2016. doi:10.1137/1.9781611974331.ch28.

4 Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025*, pages 2005–2039, 2025. doi:10.1137/1.9781611978322.63.

5 Sepehr Assadi, Gary Hoppenworth, and Nicole Wein. Covering approximate shortest paths with dags. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025*, pages 2269–2280. Association for Computing Machinery, 2025. doi:10.1145/3717823.3718298.

6 Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Toward tight approximation bounds for graph diameter and eccentricities. *SIAM Journal on Computing*, 50(4):1155–1199, 2021. doi:10.1137/18M1226737.

7 Thiago Bergamaschi, Monika Henzinger, Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. New techniques and fine-grained hardness for dynamic near-additive spanners. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1836–1855. SIAM, 2021. doi:10.1137/1.9781611976465.110.

8 Aaron Berger, Jenny Kaufmann, and Virginia Vassilevska Williams. Approximating min-diameter: Standard and bichromatic. In *31st Annual European Symposium on Algorithms, ESA 2023*, pages 17:1–17:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.17.

9 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. *SIAM J. Comput.*, 41(6):1380–1425, 2012. doi:10.1137/110826655.

10 Greg Bodwin and Gary Hoppenworth. Folklore sampling is optimal for exact hopsets: Confirming the $\sqrt{n}$ barrier. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 701–720. IEEE, 2023. doi:10.1109/FOCS57990.2023.00046.

11 Édouard Bonnet. 4 vs 7 Sparse Undirected Unweighted Diameter is SETH-Hard at Time $n^{4/3}$. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, pages 34:1–34:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.34.

12 Karl Bringmann, Marvin Künnemann, and Karol Wegrzycki. Approximating apsp without scaling: equivalence of approximate min-plus and exact min-max. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*, pages 943–954. ACM, 2019. doi:10.1145/3313276.3316373.

13 Manuel Cáceres. Minimum Chain Cover in Almost Linear Time. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023*, volume 261, pages 31:1–31:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.31.

14 Parinya Chalermsook, Yonggang Jiang, Sagnik Mukhopadhyay, and Danupon Nanongkai. Shortcuts and transitive-closure spanners approximation. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026*. SIAM, 2026. doi:10.1137/1.9781611978971.129.

15 Shiri Chechik and Tianyi Zhang. Constant approximation of min-distances in near-linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 896–906. IEEE, 2022. doi:10.1109/FOCS54457.2022.00089.

16 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015. doi:10.1007/978-3-319-21275-3.

17 Mina Dalirrooyfard, Ce Jin, Virginia Vassilevska Williams, and Nicole Wein. Approximation Algorithms and Hardness for n-Pairs Shortest Paths and All-Nodes Shortest Cycles. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science, FOCS 2022*. IEEE, 2022. doi:10.1109/FOCS54457.2022.00034.

18 Mina Dalirrooyfard and Jenny Kaufmann. Approximation Algorithms for Min-Distance Problems in DAGs. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, pages 60:1–60:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.60.

19 Mina Dalirrooyfard, Virginia Vassilevska Williams, Nikhil Vyas, Nicole Wein, Yinzhan Xu, and Yuancheng Yu. Approximation algorithms for min-distance problems. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019*, pages 46:1–46:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.46.

20 Sally Dong and Guanghao Ye. Faster Min-Cost Flow and Approximate Tree Decomposition on Bounded Treewidth Graphs. In *32nd Annual European Symposium on Algorithms, ESA 2024*, pages 49:1–49:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.49.

21 Guillaume Ducoffe. The diameter of at-free graphs. *Journal of Graph Theory*, 99(4):594–614, 2022. doi:10.1002/jgt.22754.

22 Guillaume Ducoffe. Eccentricity queries and beyond using hub labels. *Theoretical Computer Science*, 930:128–141, 2022. doi:10.1016/j.tcs.2022.07.017.

23 Arnold Filtser. Stochastic embedding of digraphs into dags. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026*. SIAM, 2026. doi:10.1137/1.9781611978971.5.

24 Jeremy T. Fineman. Nearly work-efficient parallel algorithm for digraph reachability. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 457–470, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3188745.3188926.

25 Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *Journal of algorithms*, 53(1):85–112, 2004. doi:10.1016/j.jalgor.2004.05.002.

26 Bernhard Haeupler, Yonggang Jiang, and Thatchaphol Saranurak. Reducing shortcut and hopset constructions to shallow graphs. In *Proceedings of the 2026 SIAM Symposium on Simplicity in Algorithms, SOSA 2026*, pages 385–393. SIAM, 2026. doi:10.1137/1.9781611978964.30.

27 C. S. Karthik and Pasin Manurangsi. On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. *Combinatorica*, 40(4):539–573, 2020. doi:10.1007/s00493-019-4113-1.

28 Ton Kloks, Dieter Kratsch, and Haiko Müller. Finding and counting small induced subgraphs efficiently. *Information Processing Letters*, 74(3):115–121, 2000. doi:10.1016/S0020-0190(00)00047-8.

29 Shimon Kogan and Merav Parter. Faster and unified algorithms for diameter reducing shortcuts and minimum chain covers. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 212–239. SIAM, 2023. doi:10.1137/1.9781611977554.CH9.

30 Shimon Kogan and Merav Parter. The algorithmic power of the greene-kleitman theorem. In *32nd Annual European Symposium on Algorithms, ESA 2024*, pages 80:1–80:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.80.

31 Shimon Kogan and Merav Parter. Giving some slack: Shortcuts and transitive closure compressions. In *32nd Annual European Symposium on Algorithms, ESA 2024*, pages 79:1–79:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.79.

32 Dieter Kratsch and Jeremy Spinrad. Between o(nm) and o(na). In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2003*, pages 709–716. SIAM, 2003.

33 Ray Li. Settling seth vs. approximate sparse directed unweighted diameter (up to (nu)nseth). In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021*, pages 1684–1696. Association for Computing Machinery, 2021. doi:10.1145/3406325.3451045.

34 Yang P. Liu, Arun Jambulapati, and Aaron Sidford. Parallel reachability in almost linear work and square root depth. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 1664–1686. IEEE, 2019. doi:10.1109/FOCS.2019.00098.

35 Jan Obdrzálek. Dag-width: connectivity measure for directed graphs. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006*, pages 814–821. ACM, 2006. doi:10.5555/1109557.1109647.

36 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of Computing, STOC 2013*, pages 515–524. ACM, 2013. doi:10.1145/2488608.2488673.

37 Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 1260–1268. Association for Computing Machinery, 2018. doi:10.1145/3188745.3188916.

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

39 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.

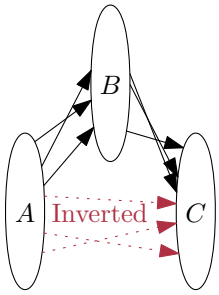
40 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. *Proceedings of the International Congress of Mathematicians (ICM 2018)*, 2019. doi:10.1142/9789813272880_0188.

41 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM*, 49(3):289–317, 2002. doi:10.1145/567112.567114.

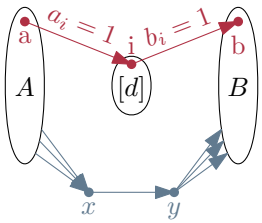
A Unweighted Graphs

In this section, we prove our results about unweighted graphs. First, we rule out small constant-factor approximations in Appendix A.1. Next, we show how to use our algorithm for weighted DAGs with small width to derandomize a classical randomized sampling algorithm in Appendix A.2. Finally, we provide our best approximation guarantee in near-linear time that achieves an $n^{1/2+o(1)}$-approximation in Appendix A.3, showing that we can go beyond a natural trade-off between approximation guarantee and running time suggested by the classical sampling algorithm.

This further deepens the connection between ReachDiam and shortcut sets and hopsets, as in the previous subsections we have seen a number of algorithms for approximating ReachDiam inspired by shortcut set algorithms and we now see the other direction: a diameter estimation algorithm yielding a shortcut set algorithm. Similar ideas also underlie the construction of almost-linear size TC-spanners for H-minor-free graphs in [9] (their result implies a comparable result for shortcut sets but not for hopsets).



(a) Lower bound construction from Theorem 17. This graph has ReachDiam 2 iff. there is a triangle in the original graph.



(b) Lower bound construction from Theorem 6. This graph has ReachDiam 3 iff. there are orthogonal vectors in $a \in A, b \in B$.

Figure 2 Our lower bound constructions for unweighted graphs.

A.1 Lower Bounds

First, we give a strong lower bound under the combinatorial BMM conjecture.

▶ **Theorem 17.** *Unless combinatorial BMM fails, there is no combinatorial algorithm that differentiates between* ReachDiam *1 and 2 in DAGs running in time* $\mathcal{O}(n^{3-\varepsilon})$.

Proof. We reduce from triangle detection in a tripartite graph G with partitions A, B, C. Create G', by directing the edges from A to B, from B to C. Take the complement of the edges between A and C and direct them from A to C. See Figure 2a for an illustration.

If there is a triangle a, b, c in G, there is a path $a \to b \to c$ in G' but no $a \to c$ edge. Hence, the diameter of G' must be 2, since this is a shortest path of length 2 and by construction, there are no longer paths. If the diameter of G' is 2, there must be a pair a, c with a path $a \to b \to c$ but no $a \to c$ edge. This triple corresponds to a triangle in G.

Thus, a combinatorial algorithm in time $\mathcal{O}(n^{3-\varepsilon})$ translates into an algorithm for triangle detection in the same time, which is impossible under the combinatorial BMM hypothesis. ◀

This immediately gives the following result.

▶ **Corollary 5.** *Unless combinatorial BMM fails, there is no combinatorial better-than-2 approximation algorithm for* ReachDiam *running in time* $\mathcal{O}(n^{3-\varepsilon})$*, even on unweighted DAGs.*

Similarly unless BMM fails, there is no better-than-2-approximation for ReachDiam *running in time* $\mathcal{O}(n^{\omega-\varepsilon})$.

Using OV, we show a different tradeoff between time and approximation factor. This result is not only more meaningful for sparse graphs, it is also based on a different, arguably more trusted, hypothesis.

▶ **Theorem 6.** *Unless OV fails, there is no better-than-3/2 approximation algorithm for* ReachDiam *running in time* $\mathcal{O}(m^{2-\varepsilon})$*, even on unweighted DAGs.*

Proof. We start with the standard embedding of an OV instance as a graph. Let $A, B \subseteq \{0,1\}^d$ with $|A|, |B| \in \Theta(n)$.

- Create $V \coloneqq A \sqcup B \sqcup [d]$.
- For $a \in A$ and $i \in [d]$, create an edge $a \to i$ iff. $a_i = 1$.
- Similarly, for $b \in B$, create an edge $i \to b$ iff. $b_i = 1$.
- To this graph, add two vertices x and y.
- Add edges from all vertices in A to x, from x to y, and from y to all vertices in B.

See Figure 2b for an illustration. Note that between any two a, b there is a path of length 2 via $[d]$ if and only if a and b are *not* orthogonal. Additionally, there is a path of length 3 from every vertex in A to every vertex in B via x and y. Clearly, this graph has a ReachDiam of 3 if there is an orthogonal pair in $A \times B$ and a ReachDiam of 2 otherwise.

Notice that the graph contains $\mathcal{O}(n)$ vertices and $\tilde{\mathcal{O}}(n)$ edges. Therefore, under the orthogonal vectors hypothesis, there can be no $\mathcal{O}(m^{2-\varepsilon})$ time algorithm that distinguishes between ReachDiam of 2 and 3, where m is the number of edges in the graph. ◀

A.2 Sampling-Based Algorithm and Its Derandomization

A simple folklore technique to approximate the diameter is based on sampling. After sampling a set S of size $\tilde{\mathcal{O}}(n/s)$ for a parameter s, with high probability, any shortest path with at least s vertices intersects S.

In this section, we first state the whole algorithm for completeness. Then, we go on to show how to replace the randomization by a chain cover that deterministically guarantees to touch every long shortest path in the graph. Then, we can use our chain-based approximation to efficiently compute ReachDiam for every path with one endpoint on a chain, which gives a similar approximation guarantee to sampling (with an additional factor 3 due to Theorem 13).

Algorithm 3 Randomized algorithm for computing ReachDiam in $\mathcal{O}(n/s \cdot m)$ time with s additive error.

Input: Directed graph $G = (V, E)$ with n vertices and parameter $s \in [n]$
Output: ReachDiam estimate $\hat{D}$ such that $D - s \leq \hat{D} \leq D$, where D is the diameter of G, whp

1. Sample $\tilde{\mathcal{O}}(n/s)$ vertices V' uniformly at random.
2. Run a forwards BFS from the vertices in V'.

Return the height of the tallest BFS tree as $\hat{D}$.

▶ **Theorem 7.** *There is an algorithm that given a parameter $k \in \mathbb{N}$, outputs an estimate $\hat{D}$ such that with high probability,* $\text{ReachDiam}(G) - k \leq \hat{D} \leq \text{ReachDiam}(G)$. *It always runs in time $\mathcal{O}(nm/k)$.*

Proof. We consider Algorithm 3. The algorithm takes time $\mathcal{O}(nm/k)$ as it requires $2n/k$ BFS runs. For the correctness, observe that as we always report the height of a BFS tree in G, we always have $\hat{D} \leq D$. For the other inequality, first observe that if $D \leq k$, the bound is trivial. Thus assume that there is a shortest path $P\colon a \rightsquigarrow b$ in G with $|P| > k$. Then, with high probability, there is a vertex $c \in V'$ in the first k vertices of P. Therefore, one of the BFS trees from c has height at least $D - k$. ◀

Let's see how to derandomize this simple sampling algorithm for DAGs. The idea is to replace the sampling by a chain cover and then use the chain-based approach of Theorem 13.

An ℓ-chain cover is similar to the minimum chain cover, but the parameter ℓ may be chosen arbitrarily to control the number of chains in the cover. As a trade-off, we then might not be able to cover all vertices, but instead get a weaker guarantee that no long chains of uncovered vertices remain. This will be enough for our use case here.

▶ **Definition 18** (ℓ-Chain Cover). *An* ℓ-chain cover *is a set of at most ℓ vertex-disjoint chains $\mathcal{C}$ in G such that for any path $P \subseteq G$, it holds that $|V(P) \setminus V(\mathcal{C})| \leq 2n/\ell$.*

Such a ℓ-chain cover always exists and can be computed in near-linear time using MCMF algorithms [13].

▶ **Lemma 19** ([13]). *There is an algorithm that computes an ℓ-chain cover of a directed graph G in time $m^{1+o(1)}$, for any ℓ.*

The core observation is now that any path either touches a chain in the cover or has length at most $2n/\ell$. Thus, if we perform our algorithm that approximates the diameter of all paths touching a chain from the previous section, we will not lose too much compared to the randomized sampling algorithm.

▶ **Theorem 8.** *There is a deterministic algorithm that given a DAG G and a parameter ℓ, outputs an estimate $\hat{D}$, such that* $(\text{ReachDiam}(G) - \ell)/3 \leq \hat{D} \leq \text{ReachDiam}(G)$ *and runs in time $\tilde{\mathcal{O}}(nm/\ell + m^{1+o(1)})$.*

Proof. Our algorithm first computes a $2n/\ell$-chain cover. Then, for each chain C in the chain cover, we run the algorithm from Theorem 13 on C. Finally, we return the maximum answer $\hat{D}$ returned for any chain.

The runtime follows immediately from Lemma 19 and Theorem 13. For the correctness, let $a, b \in V$ be a pair of vertices with $d(a, b) = \text{ReachDiam}(G)$. If $d(a, b) \leq \ell$, the bound holds trivially. Otherwise, a shortest path from a to b must intersect at least one chain from the chain cover. Let C be a first such chain on a shortest path from a to b and let v be the vertex where they intersect first. Then, by Theorem 13 we get $3\hat{D} \geq d(v, b) \geq d(a, b) - \ell$. Furthermore, we only output valid distances, so the guarantees are as claimed. ◀

A.3 Pivot-based Sampling

Next, we present a near-linear-time approximation algorithm for ReachDiam. It provides a $n^{1/2+o(1)}$ multiplicative approximation factor. Note that this is better than the $\Theta(k)$ to $\Theta(mn/k)$ trade-off shown by the (additive) approximation algorithms from the previous subsection. To achieve this, we can use a pivot-based approach pioneered by Jeremy Fineman in 2017 [24] and improved by Arun Jambulapati, Yang P. Jambulapati and Aaron Bernstein in 2019 [34] designed for computing shortcut sets, modifying it to approximate ReachDiam. While their overall goal is to compute a shortcut set in parallel, they give a sequential routine to compute a shortcut set in $\tilde{\mathcal{O}}(m)$ time. They are interested in computing a directed shortcut set with near-linear work and low depth. We see that their shortcut set quality parameter will map to our approximation guarantee. That is, the following theorem holds about Algorithm 4.

▶ **Theorem 9.** *There is an algorithm computing a $n^{1/2+o(1)}$-approximation of* ReachDiam *in $\tilde{\mathcal{O}}(m)$ time.*

The proof of this theorem can be seen as adaptation of Theorem 3 of [34]. Our stretch analysis maps to their hopbound analysis, but we need to whitebox it and show how each recursive call will affect the stretch.

For simplicity of presentation, we have left out how the shortcuts are added as it does not improve recursive behavior or the sequential running time.

Running Time. By comparing to the original algorithm, it is easy to verify that we do not perform extra work (except tracking the deepest BFS performed, which is straightforward in the presented recursion).

Algorithm 4 Algorithm of [34] adapted for ReachDiam.

Input: Directed graph $G = (V, E)$, recursion level r
Output: ReachDiam estimate $\hat{D}$, where D is the diameter of G, whp

1. Base case: If the graph has a single vertex, return 0
2. Sample vertices S from V with probability $p \in \tilde{\mathcal{O}}((\log n)^{r+1}/n)$ uar. Here n is the original number of vertices at recursion level 0.
3. Run forwards and backwards BFS from the vertices in S.
4. Each time, remove all vertices that are found in both the forward and backwards BFS from the same vertex in S.
5. Partition the remaining vertices into sets $V_1, \ldots$ s.t. all the vertices in each V_i are reachable forwards and reachable backwards from the exact same vertices from S.
6. Recursively call this algorithm on the graphs $G[V_1], \ldots$, increasing r by one.

Report the largest distance seen at any recursion level as the diameter estimate $\hat{D}$.

Correctness. Verifying that this algorithm correctly approximates the diameter requires a closer look. As part of their shortcut set quality analysis, [34] establish the following lemma, which we will rely on to establish out approximation guarantee.

▶ **Lemma 20** (Inductive application of Lemma 4.4 in [34]). *Let P be a path in G. Then, whp, we can partition P into $n^{1/2+o(1)}$ subpaths $P_1, \ldots, P_\ell$ such that for each P_i there is a recursive application of Algorithm 4 where there is a vertex u in its set S such that $(P_i)_1$ can reach u and u can reach $(P_i)_{|P_i|}$.*

The following observation follows from the triangle inequality.

▶ **Observation 21.** *Let $P = v_1, \ldots, v_{|P|}$ be a shortest path and let u be a vertex such that v_1 can reach u and u can reach $v_{|P|}$. Then $\max(d(v_1, u), d(u, v_{|P|})) \geq d(v_1, v_{|P|})/2$.*

To show that the algorithm is useful for diameter estimation, we need to examine how the recursive splitting will affect the distances between any two vertices. Towards this, the following lemma is key.

▶ **Lemma 22.** *Let u and v s.t. u can reach v. If u and v get assigned to the same subinstance by Algorithm 4, then so do all vertices between u and v (i.e., all vertices on a u-v path).*

Proof. Assume that is not the case for some vertex w in between u and v. Then there is a pivot vertex $s \in S$ such that the reachability between s and w differs from the reachability between s and u and v (which is the same).

Assume that w can be reached from s but u and v cannot. Since w is in between u and v, w can reach v and thus v can be reached from s, which contradicts our assumption. The other three cases are (1) w can reach s, but u and v cannot, (2) u and v can be reached from s but w cannot, and (3) u and v can reach s, but w cannot. These are analogous. ◀

This allows us to argue how the distances between vertices change when we create the subinstances in the algorithm. In particular, we have the following.

▶ **Corollary 23.** *Let G' be an arbitrary subinstance created in Algorithm 4 and G be the original graph. Then, for all $u, v \in V(G')$, we have $d_{G'}(u, v) = d_G(u, v)$.*

That is, if two vertices get assigned to the same subinstance, their distance is preserved. Thus, if we think about two different levels of recursion, some distances in the deeper level are ∞ because the vertices were disconnected, but all remaining non-infinite distances are the same as in the original graph. Now, we have all pieces in hand to prove the our theorem.

Proof of Theorem 9. Applying Lemma 20 to a diameter path P, we learn that whp P can be split into at most $n^{1/2+o(1)}$ subpaths such that each subpath is fully contained in subinstances of G' such that one of its pivots can reach the start and the end of the respective subpath. By the pigeon hole principle, one of these subpaths must have at least a $1/n^{1/2+o(1)}$ fraction of the length of the path. By Corollary 23, the length of this subpath is preserved in this subinstance. Applying Observation 21 to the pivot in the respective subinstance that can be reached by the start and reach the end of this subpath, yields the desired approximation guarantee as in the algorithm we will run BFS from it. ◀