


Approximation Algorithms for Min-Distance Problems in DAGs

Mina Dalirrooyfard ✉

MIT, Cambridge, MA, USA

Jenny Kaufmann ✉ 

Harvard University, Cambridge, MA, USA

Abstract

Graph parameters such as the diameter, radius, and vertex eccentricities are not defined in a useful way in Directed Acyclic Graphs (DAGs) using the standard measure of distance, since for any two nodes, there is no path between them in one of the two directions. So it is natural to consider the distance between two nodes as the length of the shortest path in the direction in which this path exists, motivating the definition of the min-distance. The min-distance between two nodes u and v is the minimum of the shortest path distances from u to v and from v to u .

As with the standard distance problems, the Strong Exponential Time Hypothesis [Impagliazzo-Paturi-Zane 2001, Calabro-Impagliazzo-Paturi 2009] leaves little hope for computing min-distance problems faster than computing All Pairs Shortest Paths, which can be solved in $\tilde{O}(mn)$ time. So it is natural to resort to approximation algorithms in $\tilde{O}(mn^{1-\epsilon})$ time for some positive ϵ . Abboud, Vassilevska W., and Wang [SODA 2016] first studied min-distance problems achieving constant factor approximation algorithms on DAGs, and Dalirrooyfard *et al* [ICALP 2019] gave the first constant factor approximation algorithms on general graphs for min-diameter, min-radius and min-eccentricities. Abboud *et al* obtained a 3-approximation algorithm for min-radius on DAGs which works in $\tilde{O}(m\sqrt{n})$ time, and showed that any $(2 - \delta)$ -approximation requires $n^{2-o(1)}$ time for any $\delta > 0$, under the Hitting Set Conjecture. We close the gap, obtaining a 2-approximation algorithm which runs in $\tilde{O}(m\sqrt{n})$ time. As the lower bound of Abboud *et al* only works for sparse DAGs, we further show that our algorithm is conditionally tight for dense DAGs using a reduction from Boolean matrix multiplication. Moreover, Abboud *et al* obtained a linear time 2-approximation algorithm for min-diameter along with a lower bound stating that any $(3/2 - \delta)$ -approximation algorithm for sparse DAGs requires $n^{2-o(1)}$ time under SETH. We close this gap for dense DAGs by obtaining a 3/2-approximation algorithm which works in $O(n^{2.350})$ time and showing that the approximation factor is unlikely to be improved within $O(n^{\omega-o(1)})$ time under the high dimensional Orthogonal Vectors Conjecture, where ω is the matrix multiplication exponent.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Fine-grained complexity, Graph algorithms, Diameter, Radius, Eccentricities

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.60

Category Track A: Algorithms, Complexity and Games

Acknowledgements We thank our advisor, Virginia Vassilevska Williams, for many helpful suggestions.

1 Introduction

Among the most fundamental graph parameters that have been extensively studied are the diameter, radius and eccentricities [16, 24, 15, 21, 5, 17, 14, 20, 7, 8, 33, 34, 12, 22, 30, 28, 13, 2, 9] (and many others). The eccentricity of a vertex v is the largest distance between v and any other vertex. The diameter is the maximum eccentricity of a vertex in the graph, thus the distance between the two farthest nodes, and the radius is the minimum eccentricity, measuring the maximum distance to the most central node.



© Mina Dalirrooyfard and Jenny Kaufmann;
licensed under Creative Commons License CC-BY 4.0

48th International Colloquium on Automata, Languages, and Programming (ICALP 2021).

Editors: Nikhil Bansal, Emanuela Merelli, and James Worrell; Article No. 60; pp. 60:1–60:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



All of these parameters depend on the definition of the distance between two nodes. In undirected graphs, the distance between two vertices is just the shortest path distance $d(\cdot, \cdot)$ between them, which is symmetric. However, in directed graphs, this standard measure of distance d is not necessarily symmetric, since for two nodes, $d(u, v)$ may not equal $d(v, u)$.

Several notions of a “symmetric” distance for directed graphs have been studied. Cowen and Wagner [18] define the *roundtrip distance*, which for two vertices u and v is just $d(u, v) + d(v, u)$. Abboud, Vassilevska W., and Wang [3] define the *max-distance*, which is $\max\{d(u, v), d(v, u)\}$, and the *min-distance*, which is $\min\{d(u, v), d(v, u)\}$.

Each of these notions of distance has a particular application [19]. In this paper, we focus on the min-distance $d_{\min}(\cdot, \cdot)$. The min-distance characterizes a quantity of real-world relevance: for instance, a patient may visit a doctor or a doctor may visit a patient, and if they are in a hurry the min-distance between them may matter. Min-distance is a particularly natural notion of distance in directed acyclic graphs (DAGs), where the standard notion of distance is infinite in at least one direction for any given pair of vertices in a DAG. For example, in a topologically ordered DAG where the edges are directed from left to right, the min-diameter is simply the largest distance $d(u, v)$ where u is to the left of v .

More formally, for a vertex $v \in V$, the *min-eccentricity* $\epsilon(v)$ is $\max_{w \in V} d_{\min}(v, w)$, or in other words, the largest min-distance between v and any other vertex. The *min-diameter* of a graph is $\max_{v \in V} \epsilon(v)$. Note that the min-diameter is the only meaningful notion of diameter for DAGs: all other notions are infinite. The *min-radius* of a graph is $\min_{v \in V} \epsilon(v)$. A *center* is a vertex whose min-eccentricity is equal to the min-radius of the graph.

All-Pairs Shortest Paths (APSP) is the problem of computing the distance between u and v for every pair of vertices $u, v \in V$. In a graph G with m edges, n vertices, and nonnegative edge weights polynomial in n , APSP can easily be computed in $\tilde{O}(mn)$ time¹, by running Dijkstra’s algorithm from every vertex². Computing eccentricities, diameter, or radius with any of the notions of distance is no harder than computing APSP.

For the standard notion of distance, under the Strong Exponential Time Hypothesis (SETH) [25, 11], there is no *truly subquadratic* time algorithm for diameter (and thus nor for eccentricities) in unweighted graphs: that is, no such algorithm runs in time $O(m^{2-\epsilon})$ for $\epsilon > 0$ [28]. This lower bound also holds for the other notions of diameter (and eccentricities) [19]. For radius, the same lower bound holds but under the Hitting Set Conjecture [3].

Since quadratic time is expensive on large graphs, we resort to approximation algorithms. Many constant factor approximation algorithms were known for all notions of diameter, eccentricities and radius, except for the min-distance notion until recently. For example, for the standard diameter and roundtrip diameter there is a folklore linear time 2-approximation algorithm, and for max-diameter and standard diameter, a conditionally tight 3/2-approximation algorithm is known in $\tilde{O}(m\sqrt{n})$ time [28].

Only recently Dalirrooyfard *et al* [19] showed constant factor approximation algorithms for min-distance problems in general graphs that run in $O(mn^{1-\epsilon})$ time for some fixed $\epsilon > 0$. More specifically, they obtained a 3-approximation algorithm for min-diameter in $\tilde{O}(m\sqrt{n})$ time, a $(3 + \delta)$ -approximation algorithm for min-radius in $\tilde{O}(m\sqrt{n}/\delta)$ time, and a $(3 + \delta)$ -approximation algorithm for min-eccentricities in $\tilde{O}(m\sqrt{n}/\delta^2)$ time, for any $\delta > 0$.

The reason it is hard to obtain approximation algorithms for min-diameter, min-radius, and min-eccentricities is that min-distance does not obey the triangle inequality. Hence the typical approaches to find algorithms that work for other notions of distance do not work for min-distance, as they crucially rely on the triangle inequality.

¹ The tilde hides polylogarithmic factors.

² Faster algorithms are known by Pettie [26] and Pettie and Ramachandran [27] for sparse graphs.

■ **Table 1** Results on min-distance problems on DAGs. The (*) marks lower bounds that are for dense DAGs. Our $(2 - \delta)$ lower bound for min-radius is based on Triangle Detection and our $(\frac{3}{2} - \delta)$ lower bound for min-diameter is based on high dimensional OV. Our k and $(k + \delta)$ -approximation algorithms are for any integer $k \geq 2$. Conditionally tight bounds are in bold.

Problem	Upper bound	Lower bound	Reference
min-diameter	2 in $O(m)$	$(\frac{3}{2} - \delta)$ needs $m^{2-o(1)}$	[3]
	$\frac{3}{2}$ in $O(n^{2.350})$ (dense, unweighted)	$(\frac{3}{2} - \delta)$ needs $n^{\omega-o(1)}$ (*)	this work
min-radius	3 in $\tilde{O}(m\sqrt{n})$	$(2 - \delta)$ needs $m^{2-o(1)}$	[3]
	2 in $\tilde{O}(\min(m\sqrt{n}, m^{2/3}n))$	$(2 - \delta)$ needs $n^{\omega-o(1)}$ (*)	this work
	k in $\tilde{O}(\min(mn^{1/k}, m^{\frac{2^k-1}{2^k-1}}n))$		this work
min-eccentri.	$3 + \delta$ in $\tilde{O}(m\sqrt{n}/\delta^2)$		[19]
	$k + \delta$ in $\tilde{O}(\min(mn^{1/k}/\delta, m^{\frac{2^k-1}{2^k-1}}n/\delta))$		this work

On the bright side, since DAGs have more structure, it is easier to find algorithms for them. The best known subquadratic time algorithm for min-diameter in DAGs is a linear time 2-approximation algorithm, and the best subquadratic time algorithm for min-radius is a 3-approximation algorithm in $\tilde{O}(m\sqrt{n})$ time [3]. However, neither of these algorithms were proven to be conditionally tight.

Previously, the only known conditional lower bounds for these problems were due to Abboud, Vassilevska W., and Wang [3]. They showed that under the Orthogonal Vectors Conjecture from fine-grained complexity (and consequently under SETH [31]), there is no $(3/2 - \delta)$ -approximation algorithm for any $\delta > 0$ for min-diameter which runs in truly subquadratic time on sparse DAGs. Moreover, under the Hitting Set Conjecture, there is no $(2 - \delta)$ -approximation algorithm for any $\delta > 0$ for min-radius which runs in truly subquadratic time on sparse DAGs.

1.1 Our results

We obtain fast algorithms for min-diameter, min-eccentricities and min-radius with improved approximation factors. Our results can be seen in Table 1.

Min-Eccentricities and Min-Radius

We obtain the first known subquadratic time $(2 + \delta)$ -approximation algorithm for min-eccentricities in DAGs for any $\delta > 0$, and the first known subquadratic time 2-approximation algorithm for min-radius in DAGs. These algorithms run in time $\tilde{O}(\min(m\sqrt{n}/\delta, m^{2/3}n)/\delta)$ and $\tilde{O}(\min(m\sqrt{n}, m^{2/3}n))$ respectively. Note that our algorithms in this section are combinatorial: they do not exploit fast matrix multiplication and are potentially practical. Our results are *conditionally optimal* in both sparse and dense graphs: For sparse graphs, if the Hitting Set Conjecture is true, then our min-radius result is tight and our min-eccentricity result is essentially tight, in the sense that no approximation factor smaller than 2 can be achieved in subquadratic time for either of these problems [3]. For dense graphs, our 2-approximation algorithm works in $\tilde{O}(n^{7/3})$ time, and we show that there is no $(2 - \delta)$ -approximation algorithm for min-radius (and hence min-eccentricities) in $O(n^{\omega-\epsilon})$ for $\epsilon > 0$, if the best algorithm for Triangle Detection runs in time $\Omega(n^{\omega-o(1)})$. Here $\omega < 2.37286$ [6] is the exponent of matrix multiplication.

More generally, we obtain a series of algorithms trading off runtime and accuracy.

► **Theorem 1.** *For integer $k \geq 2$ and every $\delta > 0$, there is a $(k + \delta)$ -approximation algorithm for min-eccentricities in DAGs which runs in $\tilde{O}(\min(mn^{1/k}/\delta, m^{2^{k-1}/(2^k-1)}n/\delta))$ time.*

For every integer $k \geq 2$, there is a k -approximation algorithm for min-radius in DAGs which runs in $\tilde{O}(\min(mn^{1/k}, m^{2^{k-1}/(2^k-1)}n))$ time.

As mentioned earlier, the case $k = 2$ gives a 2-approximation algorithm for min-radius running in time $\tilde{O}(\min(m\sqrt{n}, m^{2/3}n))$. For $m = \tilde{O}(n^{1.5})$, this matches the runtime and improves the approximation factor of the previous best known algorithm for this problem (from [3]). For $m = \omega(n^{1.5+o(1)})$, it improves both the approximation factor and the runtime.

Our min-eccentricity $(2 + \delta)$ -approximation algorithm borrows a key idea from the 3-approximation algorithm of [3] and combines it with a new binary search technique. The idea is to partition the DAG into intervals and do local APSP searches to find local paths, then combine these local paths with “outer” paths to guarantee a low enough min-distance to any vertex in the graph. In [3], these outer paths were found by using a clever choice of intervals; our algorithm instead applies binary search to find sets which can be used as jumping-off points for the outer paths, allowing us to shorten the lengths of these paths and also allowing us to approximate all min-eccentricities, not only min-radius. Our $(k + \delta)$ -approximation algorithm is achieved by recursively running our approximation algorithm on the intervals instead of running local APSP, which allows us to improve the runtime.

For sparse graphs, Abboud, Vassilevska W., and Wang [3] already showed that a $(2 - \delta)$ -approximation for min-radius needs $\Omega(m^{2-o(1)})$ time under the Hitting Set Conjecture, so our 2-approximation algorithm is conditionally tight for sparse graphs. We show that the approximation factor of our algorithm is conditionally tight for the dense case as well by reducing Triangle Detection to $(2 - \delta)$ -approximation of min-radius for any $\delta > 0$. The best running time for Triangle Detection in n -node graphs is conjectured to be $\Omega(n^{\omega-o(1)})$ by many papers (see for example [1, 10]), where $\omega < 2.37286$ [6] is the exponent of fast matrix multiplication. Note that, since $m = O(n^2)$, our algorithm runs in $\tilde{O}(n^{7/3})$ time, which is faster than $O(n^\omega)$ for the current best bound on ω . Since the algorithm of Theorem 1 is combinatorial, if we restrict to combinatorial algorithms then there is no *truly subcubic* (meaning $O(n^{3-\epsilon})$ for $\epsilon > 0$) time $(2 - \delta)$ -approximation algorithm for min-radius provided that there is no truly subcubic time combinatorial algorithm for Boolean matrix multiplication (BMM). This is because BMM and Triangle Detection are subcubic equivalent [32]. Note that our reduction graph in Theorem 2 is an unweighted DAG.

► **Theorem 2.** *If there is a $T(n, m)$ -time algorithm for $(2 - \delta)$ -approximation of min-radius in $O(n)$ -node $\tilde{O}(m)$ -edge DAGs for some $\delta > 0$, then there is an $\tilde{O}(T(n, m) + m)$ -time algorithm for Triangle Detection on graphs with n nodes and m edges.*

► **Corollary 3.** *Assuming the best algorithm for Triangle Detection runs in time $\Omega(n^{\omega-o(1)})$, there is no algorithm for $(2 - \delta)$ -approximation of min-radius in n -node dense DAGs that runs in time $O(n^{\omega-\epsilon})$ for any $\delta, \epsilon > 0$.*

Moreover, there is no $O(n^{3-\epsilon})$ -time combinatorial algorithm for $(2 - \delta)$ -approximation of min-radius in n -node dense DAGs with $\epsilon, \delta > 0$ if there is no $O(n^{3-\epsilon'})$ -time combinatorial algorithm for BMM with $\epsilon' > 0$.

Improving the running time using Fast Matrix Multiplication

In DAGs with small integer edge weights, we further improve the running times for all k in Theorem 1 by applying a result of Zwick in [36] on the runtime of APSP in such graphs. We describe our result in more detail in Section 2. In particular, in DAGs with constant integer edge weights, including unweighted DAGs, our result in the case $k = 2$ is as follows:

► **Theorem 4.** *For every $\delta > 0$, there is an $\tilde{O}(\min(m\sqrt{n}/\delta, m^{0.605}n/\delta))$ -time $(2 + \delta)$ -approximation algorithm for min-eccentricities in DAGs with constant integer edge weights.*

There is an $\tilde{O}(\min(m\sqrt{n}, m^{0.605}n))$ -time 2-approximation algorithm for min-radius in DAGs with constant integer edge weights.

Min-Diameter

We obtain a 3/2-approximation algorithm for min-diameter in unweighted DAGs, where the approximation factor is conditionally optimal in dense graphs. Specifically, our algorithm improves on the standard APSP runtime for any graph with $m = \omega(n^{1+o(1)})$ edges. This is the first known 3/2-approximation algorithm for min-diameter in dense DAGs that runs faster than the best constant factor approximation algorithm for APSP, which runs in $\tilde{O}(n^\omega)$ time in unweighted directed graphs [36].

► **Theorem 5.** *There is an $O(m^{0.414}n^{1.522} + n^{2+o(1)})$ -time 3/2-approximation algorithm for min-diameter in unweighted DAGs.*

This algorithm relies on the sparse matrix multiplication algorithm of Yuster and Zwick [35]. In dense graphs with $m = O(n^2)$, its runtime is $O(n^{2.350})$. In relatively sparse graphs, with $m = O(n^{1.154+o(1)})$, the second term dominates, so the runtime is $O(n^{2+o(1)})$.

Our techniques, which mix known diameter techniques with sparse matrix multiplication, are informally as follows: We first construct a covering set, which will intersect any sufficiently large set. We run BFS from all vertices in the covering set, and check whether any min-distances found were large. If not, then for each vertex u , we will define a set of vertices that are relatively “close” to u on its right; if this set is large it will intersect the covering set, allowing us to find paths from u to some vertices to its right, using a “close” vertex in the covering set as a jumping-off point. The remaining vertices w , for which this method did not construct a $u \rightarrow w$ path, must have the property that any $u \rightarrow w$ path must intersect a relatively small subset of the set of vertices “close” to u (note that this set may have been small to begin with, in which case we can skip the previous step). Symmetrically, for each vertex w we can construct the corresponding relatively small subset of vertices “close” to w on its left, and then to bound the min-distance between u and w we check whether these two small subsets share a vertex in common. We use sparse matrix multiplication to detect this set intersection.

The conditional lower bound of [3] says that if the Orthogonal Vectors Conjecture is true then min-diameter cannot be $(3/2 - \delta)$ -approximated in truly subquadratic time in sparse graphs. There is no known 3/2-approximation algorithm for min-diameter on DAGs that works faster than APSP, neither for dense graphs nor for sparse graphs. So the question is: Is 3/2 the right bound for inapproximability of min-diameter in DAGs? We answer this question in the affirmative for dense DAGs. Theorem 5 gives the first 3/2-approximation algorithm that works faster than APSP, and it is optimal conditioned on *high dimensional OV* using the same reduction as [3]. High dimensional OV can be used for obtaining lower bounds for dense graphs. In high dimensional OV, the dimension of the vectors can be as big as $O(n)$, and using a simple reduction to Boolean matrix multiplication, the best known algorithm for it is in time $O(n^\omega)$.

High dimensional OV gives a conditional lower bound of $\Omega(n^{\omega-o(1)})$ time for $(3/2 - \delta)$ -approximation of min-diameter for any $\delta > 0$. Our algorithm gives an upper bound of $O(n^{2.350})$ for $m = \Theta(n^2)$, which is faster than $O(n^\omega)$ for the current best bound on ω . We note while we provide tight results for dense DAGs, the gap between the lower bound and upper bound for computing min-diameter on sparse DAGs is still open.

1.2 Preliminaries

All graphs in this paper are directed graphs. Given a graph G , n denotes the number of vertices and m denotes the number of edges. We will assume $m \geq n - 1$ since otherwise all min-eccentricities are infinite, a case that is easily checked. All edge weights are assumed to be nonnegative and polynomial in n ; if w_{max} is the maximum edge weight and w_{min} is the minimum edge weight, we let $M = \max\{w_{max}, 1/w_{min}\}$. We write $G[S]$ to denote the subgraph of G induced by vertex set S . For a vertex v , we write $N_D^{\text{in}}(v)$ (respectively, $N_D^{\text{out}}(v)$) to denote the set of vertices u such that $d(u, v) \leq D$ (respectively, $d(v, u) \leq D$).

For $v \in V$ and $W \subseteq V$, we define $d_{\min}(W, v) = d_{\min}(v, W)$ as $\min_{w \in W} d_{\min}(v, w)$, and we define the min-eccentricity of W as $\epsilon(W) = \max_{v \in V} d_{\min}(W, v)$.

Given two sets $U, W \subseteq V$, if every $u \in U$ appears prior to (respectively, after) every $w \in W$ in a topological ordering of the vertices of G , we say that U is the left (respectively, right) of W with respect to the topological ordering. When U or W consists of a single vertex $\{x\}$, we omit the brackets. If $W \subseteq U \subseteq V$, we denote the subset of vertices in U that lie to the left (right) of W by $L_U(W)$ (respectively, $R_U(W)$). If $U = V$, we omit the subscript. A vertex set W is called *topologically consecutive* with respect to a topological ordering if its vertices are consecutive; i.e., if $W = V \setminus (L(W) \cup R(W))$. In general, the relevant topological ordering will be clear, and we will omit reference to it.

Let $\omega(1, r, 1)$ be the exponent of the runtime of multiplying $n \times n^r$ by $n^r \times n$ matrices. Let $\omega = \omega(1, 1, 1)$ be the square matrix multiplication exponent. [6] showed that $\omega < 2.37286$.

For specifying lower bounds, we use the following problems with their corresponding running time conjectures.

Orthogonal Vectors (OV)

Given two lists A, B of n d -dimensional Boolean vectors, determine whether there are vectors $a \in A$ and $b \in B$ such that a and b are *orthogonal*; i.e. there is no $i \in [d]$ such that the i th bits of both a and b are 1. When $d = \Omega(\log n)$, the *OV Conjecture* [31] says that there is no algorithm that can solve the OV problem in time $O(n^{2-\epsilon})$ for any fixed $\epsilon > 0$. The OV Conjecture is implied by the Strong Exponential Time Hypothesis (SETH) [31].

High Dimensional Orthogonal Vectors

In high dimensional OV, the dimension d can be as high as $O(n)$. There is a simple reduction from high dimensional OV to matrix multiplication: Given two lists $A = \{a_1, \dots, a_n\}$, $B = \{b_1, \dots, b_n\}$ of d -dimensional Boolean vectors, let M and N be two $n \times d$ and $d \times n$ Boolean matrices, where $M[i, j] = 1$ if a_i is 1 in bit j , and $N[j, k] = 1$ if b_k is 1 in bit j , for $j = 1, \dots, d$ and $i, k = 1, \dots, n$. If MN has a zero entry, the vector pair corresponding to that entry are orthogonal. This gives a $O(n^\omega)$ algorithm for high dimensional OV, and there are no faster algorithms known for it up to polylogarithmic factors. Moreover, OV is equivalent to the problem of distinguishing diameter 2 vs 3 [28], and so high dimensional OV is equivalent to distinguishing diameter 2 vs 3 in dense graphs. A well-known open problem is whether diameter 2 vs 3 can be solved faster than matrix multiplication (see for example [5]). Hence, it is conjectured that high dimensional OV cannot be solved in $O(n^{\omega-\epsilon})$ time for any $\epsilon > 0$.

Hitting Set (HS)

Given two lists $A, B \in \{0, 1\}^d$, determine whether there is a vector $a \in A$ that is not orthogonal to any vector $b \in B$. When $d = \Omega(\log n)$, the *Hitting Set Conjecture* [3] says that there is no algorithm that can solve the Hitting Set problem in time $O(n^{2-\delta})$ for any fixed $\delta > 0$.

Boolean Matrix Multiplication (BMM)

We abbreviate multiplying two Boolean $n \times n$ matrices over the (AND, OR)-semiring by BMM. It is conjectured that there is no combinatorial algorithm solving BMM in $O(n^{3-\epsilon})$ time for any fixed $\epsilon > 0$, and the best algebraic algorithm for it is in $O(n^{\omega+o(1)})$ time for $\omega < 2.37286$ [6].

Triangle Detection [32]

Given a tripartite graph $G(A, B, C, E)$ where A, B and C are the three parts of the vertex set and E is the edge set, determine if there are $a \in A, b \in B$, and $c \in C$ such that abc is a triangle. Vassilevska W. and Williams [32] showed that considering only combinatorial algorithms, Triangle Detection and BMM are subcubic equivalent, meaning that a truly subcubic combinatorial algorithm in one results in a truly subcubic combinatorial algorithm in the other. Moreover, the best (algebraic) algorithm for Triangle Detection is through BMM. Thus the best running time for Triangle Detection is $O(n^\omega)$, and it is conjectured (see for example [1, 10]) that there is no algorithm faster than $O(n^\omega)$ for detecting a triangle.

2 Min-Eccentricities and Min-Radius

We present two different versions of our min-eccentricity and min-radius approximation algorithms, one which works in general weighted DAGs and is combinatorial and one with a lower runtime upper bound which only works in DAGs with small integer edge weights. The algorithms are identical except in how they compute APSP; the former computes APSP in the standard combinatorial way, while the latter uses Zwick's fast APSP algorithm for graphs with small integer edge weights. Here, $\mu(t)$ is the value satisfying $\omega(1, \mu(t), 1) = 1 + 2\mu(t) - t$.

► **Theorem 6** ([36]). *APSP can be computed in $O(n^{2+\mu(t)})$ time in directed graphs with integer edge weights bounded by n^t , where $t < 3 - \omega$.*

Both versions of our algorithms use a common technique to compute min-distances to and from a vertex set. Given a graph G and a vertex set $W \subseteq V$, we construct a graph G' by adding a vertex y and adding weight-0 edges (w, y) for all $w \in W$. We then run Dijkstra into y in G' . We refer to this procedure as *running Dijkstra into W* . The symmetric procedure, in which the weight-0 edges point out of an added vertex y' and we run Dijkstra out of y' , will be referred to as *running Dijkstra out of W* . Then for $x \in V$, $d_{\min}(x, W) = \min(d(x, y), d(y', x))$, a value which we can now compute. We added $|W|$ edges and ran Dijkstra in G' , so in total the procedure takes time $O(|W| + m \log n) = O(m \log n)$.

Our min-eccentricity and min-radius approximation algorithms will be based on the following proposition. Let $c_k(\tau) = \frac{2^{k-2}(1+\tau)}{2^{k-1}(1+\tau)-\tau}$.

► **Proposition 7.** *For any $k \geq 2$, there is an $O(\min(mn^{1/k} \log^2 n, m^{2^{k-1}/(2^k-1)} n \log^2 n))$ -time algorithm which takes as input a DAG G and a parameter r , and certifies for each vertex v that $\epsilon(v) > r$ or that $\epsilon(v) \leq kr$.*

In DAGs with integer edge weights bounded by n^t , where $t < 3 - \omega$, there is a version of this algorithm which runs in $O(\min(mn^{1/k} \log^2 n, m^{c_k(\mu(t))} n \log^2 n))$ -time.

In [23], Le Gall and Urrutia showed that $\mu = \mu(0) < 0.529$. Thus in DAGs with constant integer edge weights (so that $t = 0$), the runtime of the algorithm of Proposition 7 is $\tilde{O}(\min(mn^{1/k}, m^{c_k(0.529)} n))$ time. When $k = 2$, $c_k(0.529) < 0.605$, leading to the special case stated in Theorem 4.

The algorithms of Proposition 7 will be described and proven correct in Subsection 2.1, and their runtimes will be analyzed in Lemma 13 in Subsection 2.2. Then by binary searching over $r \in [0, Mn]$, these algorithms can be used to obtain the min-eccentricity approximation algorithms of Theorems 8 and 9 and the min-radius approximation algorithms of Theorems 10 and 11.

► **Theorem 8.** *Let $k \geq 2$ be an integer. For any $\delta > 0$, there is an $\tilde{O}(\min(mn^{1/k}/\delta, m^{2^{k-1}/(2^k-1)}n/\delta))$ -time algorithm which, given a DAG G , outputs for every vertex $v \in V$ an estimate $\epsilon'(v)$ such that $\epsilon(v) \leq \epsilon'(v) < (k + \delta)\epsilon(v)$.*

► **Theorem 9.** *Let $k \geq 2$ be an integer. For any $\delta > 0$, there is an $\tilde{O}(\min(mn^{1/k}/\delta, m^{c_k(\mu(t))}n/\delta))$ time algorithm which, given a DAG G with integer edge weights bounded by n^t for $t < 3 - \omega$, outputs for every vertex $v \in V$ an estimate $\epsilon'(v)$ such that $\epsilon(v) \leq \epsilon'(v) < (k + \delta)\epsilon(v)$.*

Proof. First we have all the vertices as “unmarked.” We do binary search in $[0, Mn]$ by starting with $r = 1$ in Proposition 7 and incrementing $r' = (1 + \delta/k)r$ at each step. At each step, we run the algorithm given in Proposition 7, and for each unmarked v that is reported as having $\epsilon(v) \leq kr$, we set $\epsilon'(v) = kr$ and mark v . At the end we set $\epsilon'(v) = \infty$ for any remaining unmarked vertices.

Suppose a vertex v was marked at the step corresponding to r . Then $r/(1 + \delta/k) < \epsilon(v) \leq kr$, so $\epsilon(v) \leq \epsilon'(v) = kr < (k + \delta)\epsilon(v)$. The binary search adds an $O(\log_{1+\delta/k} Mn) = O((\log Mn)/\delta)$ factor to the runtime. Since $\log Mn$ is polylogarithmic in n , this gives the time bounds stated. ◀

► **Theorem 10.** *Let $k \geq 2$ be an integer. There is an $\tilde{O}(\min(mn^{1/k}, m^{2^{k-1}/(2^k-1)}n))$ -time algorithm which, given a DAG G , outputs an approximation R' such that if R is the min-radius of G , $R \leq R' < kR$.*

► **Theorem 11.** *Let $k \geq 2$ be an integer. There is an $\tilde{O}(\min(mn^{1/k}, m^{c_k(\mu(t))}n))$ -time algorithm which, given a DAG G with integer edge weights bounded by n^t for $t < 3 - \omega$, outputs an approximation R' such that if R is the min-radius of G , $R \leq R' < kR$.*

Proof. We do binary search in $[0, Mn]$, running the algorithm given by Proposition 7 at each step as follows: We keep two numbers A_i and B_i at step i which are the lower bound and upper bound to the min-radius R . At step 1 we have $A_1 = 0$ and $B_1 = Mn$. At step i , we have A_i, B_i such that $A_i < R \leq B_i$. Let $C_i = B_i - kA_i$. If C_i is smaller than the minimum positive edge weight, then any path of length at most B_i must have length at most kA_i , so in this case we terminate the binary search and let $R' = kA_i$. We now have $R \leq R' < kR$ as desired.

If C_i is not smaller than the minimum positive edge weight, let $r = A_i + \frac{C_i}{k+1}$, and run the algorithm given by Proposition 7. If the algorithm reports that there is a vertex v with $\epsilon(v) < kr$, then let $A_{i+1} = A_i$ and $B_{i+1} = kr = kA_i + \frac{k}{k+1}C_i$, as we have the min-radius is between A_{i+1} and B_{i+1} . Note that in this case $C_{i+1} = B_{i+1} - kA_{i+1} = \frac{k}{k+1}C_i$. Otherwise, if the algorithm reports that every vertex has $\epsilon(v) \geq r$, then the min-radius is at least $A_{i+1} := r = A_i + \frac{C_i}{k+1}$ and is less than $B_{i+1} := B_i$. In this case $C_{i+1} = B_i - k(A_i + \frac{C_i}{k+1}) = \frac{k}{k+1}C_i$. Thus, at each step, the size of C_i shrinks by a factor of $\frac{k}{k+1}$. Hence, for constant k , the algorithm will in $O(\log Mn)$ steps find bounds A_i, B_i such that C_i is smaller than the minimum positive edge weight. ◀

2.1 Algorithm Description and Correctness

We now describe and prove the correctness of the algorithm of Proposition 7 by induction on k . For convenience, we use $k = 1$ as a base case; in this case we simply run an APSP computation. Our algorithm for $k > 1$ is as follows.

First, topologically sort the vertices and partition them into p consecutive sets W_1, \dots, W_p of size $|W_i| = n/p$. The runtime-minimizing value of p will be chosen later.

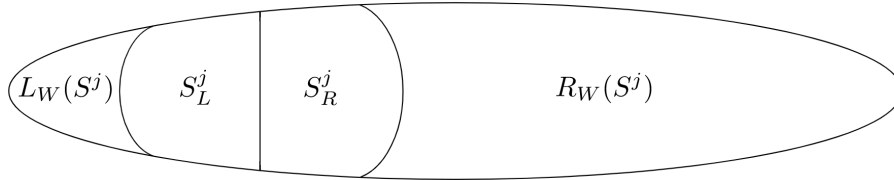
For each i , run Dijkstra to and from W_i . If $\epsilon(W_i) > r$, then we can report $\epsilon(w) > r$ for all $w \in W_i$. Otherwise, $\epsilon(W_i) \leq r$. In this case, we will apply Claim 12, below, twice. Recall that for $S \subseteq W \subseteq V$, $L_W(S)$ is the set of vertices in W that are to the left of all vertices in S in the topological ordering.

▷ **Claim 12.** Let $W \subseteq V$ be a topologically consecutive subset of a topologically ordered DAG G , and let r be a parameter such that $\epsilon(W) \leq r$. In $O(m \log^2 n)$ time, one can find a nonempty topologically consecutive subset $S \subseteq W$ such that:

- (a) $\epsilon(S) \leq r$.
- (b) If $w \in L_W(S)$, $\epsilon(w) > r$.
- (c) If $|S| > 1$, all vertices $s \in S$ satisfy $\epsilon(s) > r$.

Proof. We will use a binary search argument to find S . We will induct on an index j . Let $S^0 = W$. Assume that $S^j \subseteq W$ is topologically consecutive, that $\epsilon(S^j) \leq r$, and that for every $w \in L_W(S^j)$, $\epsilon(w) > r$. These all hold for $j = 0$. If $S^j = \{s\}$ consists of a single vertex, let $S = S^j$; then we are done.

Otherwise, let S_L^j be the subset of S^j containing its first $|S^j|/2$ vertices in the topological ordering and let $S_R^j = S^j \setminus S_L^j$. So S_L^j and S_R^j are the left and right halves of S^j , respectively; hence both S_L^j and S_R^j are topologically consecutive. See Figure 1.



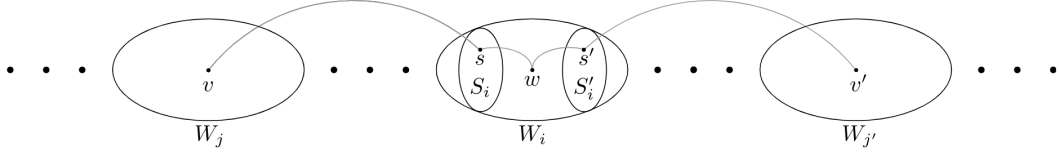
■ **Figure 1** S^j is partitioned into two halves, S_L^j and S_R^j .

Run Dijkstra from S_L^j and from S_R^j . If either of these sets has min-eccentricity at most r , we will continue the induction: If $\epsilon(S_L^j) \leq r$, we let $S^{j+1} = S_L^j$. Then $L_W(S^{j+1}) = L_W(S^j)$, so for every $w \in L_W(S^{j+1})$, $\epsilon(w) > r$. Alternatively, if $\epsilon(S_L^j) > r$ but $\epsilon(S_R^j) \leq r$, we let $S^{j+1} = S_R^j$. Then $L_W(S^{j+1}) = L_W(S^j) \cup S_L^j$, so for every $w \in L_W(S^{j+1})$, $\epsilon(w) > r$.

Otherwise, $\epsilon(S_L^j) > r$ and $\epsilon(S_R^j) > r$. In this case we halt the induction and let $S = S^j$. Every $w \in L_W(S^j) \cup S^j$ satisfies $\epsilon(w) > r$, so S has the properties desired.

At each step, the size of the set S^j halves, so there are at most $\log |W| \leq \log n$ iterations. In each iteration, we perform a constant number of Dijkstras, so the runtime is $O(m \log^2 n)$. ◁

For each i such that $\epsilon(W_i) \leq r$, let S_i be the subset constructed by applying Claim 12 to the set $W = W_i$. For each $w \in L_{W_i}(S_i)$, we report that $\epsilon(w) > r$; this holds by Claim 12b. If S_i consists of a single vertex $\{s\}$, we can determine that for any $v \in L(W_i)$, $d_{\min}(v, s) \leq \epsilon(s) \leq r \leq kr$, by Claim 12a. Otherwise, $|S_i| > 1$, so we report that $\epsilon(s) > r$ for all $s \in S_i$; this holds by Claim 12c.



■ **Figure 2** A representation of the $v \rightarrow w$ and $w \rightarrow v'$ paths, via the sets S_i and S'_i constructed with Claim 12. The outer subpaths are of length $\leq r$, and the inner subpaths are of length $\leq (k-1)r$.

Using a recursive application of our algorithm to the graph $G_i = G[W_i]$, we can certify, for every vertex $w \in W_i$, that $\epsilon_{G_i}(w) > r$ or that $\epsilon_{G_i}(w) \leq (k-1)r$. Consider any $w \in R_{W_i}(S_i)$. If we determined that $\epsilon_{G_i}(w) > r$, we report that $\epsilon(w) > r$; this holds since $\epsilon(w) \geq \epsilon_{G_i}(w)$. Otherwise, consider any $v \in L(W_i)$. Since $\epsilon(S_i) \leq r$, there is some $s \in S_i$ such that $d_{\min}(v, s) = d(v, s) \leq r$. Then since $\epsilon_{G_i}(w) \leq (k-1)r$ and since w is to the right of s in the topological ordering, we have $d_{\min}(v, w) \leq d(v, s) + d(s, w) \leq r + (k-1)r = kr$. See Figure 2.

Thus, our algorithm has certified for each $w \in W_i$ that $\epsilon(w) > r$ or that $d_{\min}(v, w) \leq kr$ for all $v \in L(W_i)$. By a symmetric argument, we can construct the set S'_i obtained by applying Claim 12 to the graph G with the edges reversed; see Figure 2. Then as above we can determine for each $w \in W_i$ that $\epsilon(w) > r$ or that $d_{\min}(w, v') \leq kr$ for all $v' \in R(W_i)$. Since W_i is a topologically consecutive set, $V \setminus W_i = L(W_i) \cup R(W_i)$. So for any $w \in W_i$, if we determine that $d_{\min}(w, v) \leq kr$ for all $v \in L(W_i)$ and for all $v' \in R(W_i)$ we report that $\epsilon(w) \leq kr$; otherwise we report $\epsilon(w) > r$.

2.2 Runtime Analysis

In this section we analyze the runtime of the algorithm of Proposition 7, and we give full descriptions of how to prove Theorems 8-11 from Proposition 7 using binary search.

Recall that $c_k(\tau) = \frac{2^{k-2}(1+\tau)}{2^{k-1}(1+\tau)-\tau}$.

► **Lemma 13.** *The algorithm of Proposition 7 runs in time $O(\min(mn^{1/k} \log^2 n, m^{2^{k-1}/(2^k-1)} n \log^2 n))$ assuming APSP computations are done in $\tilde{O}(mn)$ time.*

On graphs with integer edge weights bounded by n^t for $t < 3 - \omega$, the algorithm runs in time $O(\min(mn^{1/k} \log^2 n, m^{m^{c_k(\mu(t))}} n \log^2 n))$, assuming APSP computations are done in $O(n^{2+\mu(t)})$ time using Zwick's fast APSP algorithm [36].

Proof. To simultaneously analyze both versions of the algorithm, our algorithm's runtime will be described in terms of a placeholder τ , such that APSP computations within the algorithm are done in $O(n^{2+\tau} \log n)$ time. To obtain the runtime bound for general weighted DAGs, we will let $\tau = 1$, and note $c_k(1) = \frac{2^{k-1}}{2^k-1}$. To obtain the runtime bound for DAGs with integer edge weights bounded by n^t for $t < 3 - \omega$, we will let $\tau = \mu(t)$.

Topologically sorting the graph takes $O(m \log n)$ time which is absorbed into the final runtime.

In order to use $k = 1$ as a base case, our inductive hypothesis will assume a slightly weaker claim about the runtime: in the inductive step for k , we will assume there is an $O(\min(mn^{1/(k-1)} \log^2 n, n^{2^{c_{k-1}(\tau)+1}} \log^2 n))$ -time algorithm which certifies for each $v \in V$ that $\epsilon(v) > r$ or that $\epsilon(v) \leq (k-1)r$. Note that $n^{2^{c_{k-1}} \tau} \geq m^{c_{k-1}}$. Then in the base case where $k = 1$, APSP takes time $O(\min(mn \log n, n^{2+\tau} \log n))$, satisfying the inductive hypothesis.

Consider $k > 1$. Running Dijkstra to and from W_i for each i takes $O(mp \log n)$. It takes time $O(mp \log^2 n)$ to apply Claim 12 twice for each i , to construct sets S_i and symmetric sets S'_i (constructed in the same way as the sets S_i but with left and right swapped, pictured in Figure 2).

We also do recursive calls of our algorithm on at most p subgraphs, induced by sets W_i . Below, we analyze the runtime of the recursive calls in two different ways, giving us two upper bounds on the algorithm's runtime.

Analysis 1. Let $m_i = |E(G[W_i])|$; then note $\sum_i m_i \leq m$. For each i , the recursive call on W_i takes time $O(m_i(n/p)^{1/(k-1)} \log^2 n)$, so in total the recursive calls take time $O(m(n/p)^{1/(k-1)} \log^2 n)$. Let $p = n^{1/k}$, so that $mp = m(n/p)^{1/(k-1)}$. Then the runtime is $O(mn^{1/k} \log^2 n)$.

Analysis 2. Since $|W_i| = n/p$, a recursive call on $G[W_i]$ takes time $O((n/p)^{2c_{k-1}(\tau)+1} \log^2 n)$. We do at most p such calls, so the total runtime of the recursive calls is $O((n/p)^{2c_{k-1}(\tau)} n \log^2 n)$. Now, we choose p so that $mp = (n/p)^{2c_{k-1}(\tau)} n$. Then $m = (n/p)^{2c_{k-1}(\tau)+1}$. Recall that $c_k(\tau) = \frac{2^{k-2}(1+\tau)}{2^{k-1}(1+\tau)-\tau}$ and note that $2c_{k-1}(\tau) + 1 = \frac{2^{k-1}(1+\tau)-\tau}{2^{k-2}(1+\tau)-\tau} = \frac{2c_{k-1}(\tau)}{c_k(\tau)}$. Thus, $m^{c_k(\tau)} = (n/p)^{2c_{k-1}(\tau)}$. So the runtime of the algorithm is $O((n/p)^{2c_{k-1}(\tau)} \cdot n \log^2 n) = O(m^{c_k(\tau)} n \log^2 n)$. Since $m = O(n^2)$, this satisfies the inductive hypothesis. ◀

2.3 Lower Bounds

In this section, using an essentially linear time reduction, we reduce Triangle Detection to $(2 - \delta)$ -approximation of min-radius.

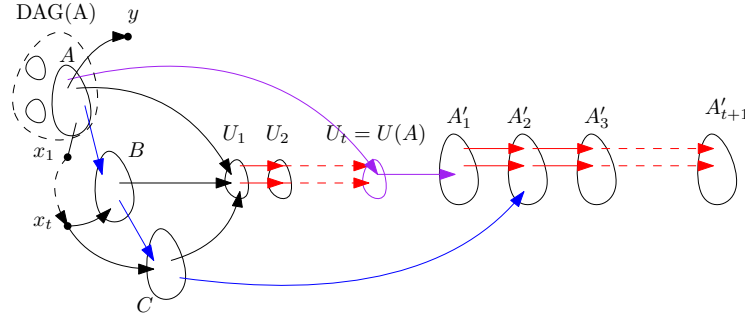
► **Reminder of Theorem 2 .** *If there is a $T(n, m)$ -time algorithm for $(2 - \delta)$ -approximation of min-radius in $O(n)$ -node $\tilde{O}(m)$ -edge DAGs for some $\delta > 0$, then there is an $\tilde{O}(T(n, m) + m)$ -time algorithm for Triangle Detection on graphs with n nodes and m edges.*

Proof. We are going to use two gadgets from previous works:

- DAG gadget [3]: Given a set X of n nodes v_1, \dots, v_n and a constant integer parameter $t \geq 2$, the gadget creates a DAG $DG_t(X)$ with at most $O(n)$ nodes and $O(n \log n)$ edges such that in the topological order of $DG_t(X)$, $v_i < v_{i+1}$, and for any two nodes of $DG_t(X)$ x, y where $x < y$ in the topological order, $d(x, y) \leq t + 1$.
- Connectivity gadget [4]: Let $X = \{v_1, \dots, v_n\}$, and let $X' = \{v'_1, \dots, v'_n\}$ be a copy of X , where both X and X' are independent sets. Then we can add a connectivity gadget $U(X)$ along with edges from X to $U(X)$ and from $U(X)$ to X' , such that $|U(X)| = O(\log n)$, for all $i \neq j$ we have $d(v_i, v'_j) = 2$, and there is no path from v_i to v'_i .

Now let $G = (A, B, C, E_G)$ be an instance of Triangle Detection, with n nodes and m edges. We create a DAG G^* such that if G has a triangle (YES case), the min-radius of G^* is $t + 1$, and if G doesn't have a triangle (NO case), the min-radius of G^* is $2t$. We let t be an integer such that $2 - \delta/2 < \frac{2t}{t+1}$, so that a fast $(2 - \delta)$ -approximation algorithm is also a fast $(\frac{2t}{t+1} - \delta/2)$ -approximation algorithm, and hence it can distinguish min-diameter $t + 1$ vs $2t$.

We define G^* as follows: G^* has A, B , and C as part of its vertex set. Let $A'_1, A'_2, \dots, A'_{t+1}$ be copies of A . Add $E_G(A, B)$ to G^* with edges directed from A to B , and add $E_G(B, C)$ with edges directed from B to C . For any $c \in C$ and $a \in A$, add an edge from c to $a' \in A'_2$ if a and c are attached in G , where a' is the copy of a in A'_2 . For each $i = 1, \dots, t$, connect the copy of a in A'_i to the copy of a in A'_{i+1} for all $a \in A$.



■ **Figure 3** Graph G^* created from the Triangle Detection instance G . Blue edges are edges in G , red edges are between two nodes that are copies of the same vertex. Purple edges are part of the connectivity gadget. Dashed lines are subpaths.

Now we add the two gadgets. Add the connectivity gadget $U(A)$ between A and A'_1 . Add two copies of $DG_t(A)$ sharing A , and denote the union of these copies by $DAG(A)$. Also add a node y , and add edges from all nodes in A to y ; this guarantees that the center of G^* must be in $DAG(A)$.

To make all nodes in A at distance $t + 1$ to A'_1 , make $t - 1$ copies of $U(A)$, U_1, \dots, U_{t-1} . For each $i = 1, \dots, t - 1$, connect the copy of u in U_i to the copy of u in U_{i+1} , for any $u \in U(A)$, where $U_t = U(A)$. Add edges from all nodes in $A \cup B \cup C$ to all nodes in U_1 .

To make all nodes in A at distance $t + 1$ to B and C , let x_1, \dots, x_t be a path of length $t - 1$. Connect all nodes of A to x_1 , and connect x_t to all nodes of $B \cup C$. See Figure 3 for the construction. Note that G^* is a DAG, with the order of sets of vertices being $DAG(A), y, x_1, \dots, x_t, B, C, U_1, \dots, U_{t-1}, U(A), A'_1, \dots, A'_{t+1}$. Moreover, $G^*[A \cup B \cup C \cup A'_2]$ has m edges corresponding to the original edges of G^* , and besides those we only added $O(n \log n)$ edges to G^* . So G^* has $O(n)$ nodes and $O(m + n \log n)$ edges.

We will show that if the Triangle Detection instance is a YES instance, then there is a node $a \in A$ such that $\epsilon(a) = t + 1$. If the Triangle Detection instance is a NO instance, then we show that for all nodes in G^* , their min-eccentricity is at least $2t$.

YES case. Let abc be a triangle in G . We show that $\epsilon(a) = t + 1$. Note that $d_{\min}(a, \bar{a}) \leq t + 1$ for all $\bar{a} \in DAG(A)$. We already know that $d(a, s) \leq t + 1$ for any $s \in B \cup C \cup \{x_1, \dots, x_t, y\}$. For any $u \in U_i$ for $i \leq t + 1$, $d(a, u) \leq t + 1$ using the path going through U_1, \dots, U_{i-1} . Since for any $z' \in A'_1$, there is a $u \in U(A)$ that has an edge to z' , we have $d(a, z') \leq t + 1$. Now for all $z' \in A'_2$ where z' is a copy of $z \in A$ and $z \neq a$, we have $d(a, z') = 3$ through $U(A)$ and A'_1 (using the edges of the connectivity gadget). For $z = a$, using the triangle edges going from A to B to C , we have that $d(a, z') = 3$. So for all $z' \in A'_2 \cup \dots \cup A'_{t+1}$, we have $d(a, z') \leq t + 1$.

NO case. Suppose that there is no triangle in G . First, note that the min-eccentricities of the vertices outside $DAG(A)$ are infinite, because there is no path between them and y . Moreover, if $z \in DAG(A) \setminus A$, it has a copy $z' \in DAG(A) \setminus A$ (in the other copy of $DG_t(A)$), and there is no path between z and z' . This is because this path must go through A , and since $DAG(A)$ consists of two copies of $DG_t(A)$ sharing A , the set of nodes in A that z has a path to (from) is exactly the same as the set of nodes in A that z' has a path to (from). So there is no $a \in A$ such that that z has a path to a and z' has a path from a .

Now it remains to compute the min-eccentricities of the vertices in A . Let $a \in A$, and let $a'_{t+1} \in A'_{t+1}$ be the copy of a . We show that $d(a, a'_{t+1}) = 2t$. Let P be a shortest path from a to a'_{t+1} . First note that any path from a to a'_{t+1} must go through $a'_2 \in A'_2$, where a'_2

is a copy of a , and we have $d(a'_2, a'_{t+1}) = t - 1$. We also know that there is no path from a to a'_2 using the edges from A to $U(A)$, because this path would need to contain a path between a and $a'_1 \in A_1$ in $G^*[A \cup U(A) \cup A'_1]$, and from the construction of the connectivity gadget there is no such path. If P does not use any $C \times A'_2$ edge, then the path must go through U_i for all i , and hence it is of length $2t$. So if the min-eccentricity of a is smaller than $2t$, the path P uses a $C \times A'_2$ edge ca'_2 for some $c \in C$. If x_1 is on the ac path, then the path goes through x_i for all i , and hence it is of length $2t$. Then x_1 is not on the path, so the ac path must go through B . In particular, there is a $b \in B$ such that $ab, bc \in E(G^*)$. Since $ca'_2 \in E(G^*)$, this implies that abc is a triangle in G , which is a contradiction. So $\epsilon(a) \geq 2t$. ◀

3 Min-diameter

Our min-diameter approximation algorithm relies on Yuster and Zwick's fast sparse matrix multiplication algorithm. Here, we define $\alpha = \max\{0 \leq r \leq 1 \mid \omega(1, r, 1) = 2\}$ and $\beta = \frac{\omega-2}{1-\alpha}$.

► **Theorem 14** ([35]). *If A and B are n by n matrices with at most l nonzero entries each, then A and B can be multiplied in $O(l^{\frac{2\beta}{\beta+1}} n^{\frac{2-\alpha\beta}{\beta+1}} + n^{2+o(1)})$ time.³*

This sparse matrix multiplication algorithm will be used to prove the following proposition.

► **Proposition 15.** *There is an $O(m^{\frac{2\beta}{3\beta+1}} n^{\frac{4\beta+2-\alpha\beta}{3\beta+1}+o(1)} + n^{2+o(1)})$ -time algorithm which, given an unweighted DAG G and a parameter D' , reports that the min-diameter D of G satisfies $D \leq \frac{3D'}{2}$ or that it satisfies $D > D'$.*

The algorithm of Proposition 15 will be described and proven to work in subsection 3.1, and its runtime will be analyzed in Lemma 18 in subsection 3.2. Then Proposition 15 allows us to obtain the min-diameter approximation algorithm given in Theorem 16 below.

► **Theorem 16.** *There is an $O(m^{\frac{2\beta}{3\beta+1}} n^{\frac{4\beta+2-\alpha\beta}{3\beta+1}+o(1)} + n^{2+o(1)})$ -time algorithm which, given an unweighted DAG G , outputs an estimate D_0 for its min-diameter D such that $D \leq D_0 < \frac{3D}{2}$.*

Proof. To obtain our approximation D_0 , we binary search over D' in $[0, n]$ by applying the algorithm of Proposition 15 logarithmically many times; note that polylogarithmic factors are $n^{o(1)}$ so they do not affect the runtime bound. Let C be the smallest value found in the binary search such that the algorithm reports that $D \leq \frac{3C}{2}$; then $D > C - 1$. Let $D_0 = \frac{3C}{2}$. Then $D \leq D_0 < \frac{3D}{2}$, as desired. ◀

Note that since $\alpha > 0.31389$ [23] and $\omega < 2.37286$ [6], we can use $\beta \simeq 0.5435$. This gives the runtime of $O(m^{0.414} n^{1.522} + n^{2+o(1)})$ stated in Theorem 5.

3.1 Algorithm Description and Correctness

Our algorithm takes as input an unweighted DAG G , an integer D' , and a parameter $\epsilon \in [0, 1]$, and reports that $D > D'$ or that $D \leq \frac{3D'}{2}$. (The runtime-minimizing value of ϵ will be determined later.)

³ To be precise, given known bounds $\alpha \geq a, \omega \leq c$, one can define $b = \frac{c-2}{1-a}$, and then equivalents of Theorem 14 hold for any such pair of values a, b , not just for the "true" values α, β . This is implicit in [35].

If at any point, a BFS finds a pair of vertices at min-distance more than D' , the algorithm reports that $D > D'$; hence in what follows we will assume that this does not occur. We initially have all pairs of vertices “unmarked,” and mark the pairs for which we know that there is a path from one to the other of length at most $\frac{3D'}{2}$.

The algorithm first takes two preliminary steps: it topologically sorts the graph, and it constructs for each vertex two topologically sorted lists, one of its in-neighbors and one of its out-neighbors.

Our algorithm will then use the greedy set cover algorithm, described in the following lemma. This lemma, and a related randomized version, are standard techniques used in graph distance algorithms (see for example [5, 28, 13, 3]). A proof may be found in [29].

► **Lemma 17.** *Let $|V| = n$, let $p = O(n)$, and let $X_1, \dots, X_p \subseteq V$ be sets of size $|X_i| \geq n^\epsilon$ for $\epsilon \in [0, 1]$. Then there is an $O(n^{1+\epsilon})$ -time algorithm which constructs a set $S \subseteq V$ of size $\tilde{O}(n^{1-\epsilon})$ such that $S \cap X_i \neq \emptyset$ for all i .*

For any $u \in V$, if $|N_{D'/2}^{\text{out}}(u)| < n^\epsilon$ let $X_u = N_{D'/2}^{\text{out}}(u)$ and otherwise let X_u be the left-most n^ϵ vertices in $N_{D'/2}^{\text{out}}(u)$. So in particular, $|X_u| \leq n^\epsilon$. We can compute X_u as follows: we maintain a list of the $\leq n^\epsilon$ left-most vertices we have found so far that are at distance $< D'/2$ from u . At each step, for each vertex in the list, we consider its left-most out-neighbor that is not yet in our set; we add the left-most such out-neighbor to the set. We halt when there are no more such out-neighbors not in our set, or after adding n^ϵ vertices to our set. Likewise, for any $w \in V$, let $Y_w = N_{D'/2}^{\text{in}}(w)$ if $|N_{D'/2}^{\text{in}}(w)| < n^\epsilon$, and otherwise let Y_w consist of the right-most n^ϵ vertices in $N_{D'/2}^{\text{in}}(w)$. We can compute the sets Y_w in a manner symmetric to how we computed the sets X_u . Then we can use Lemma 17 to construct a set S of size $\tilde{O}(n^{1-\epsilon})$ such that for all u having $|N_{D'/2}^{\text{out}}(u)| \geq n^\epsilon$, $S \cap X_u$ is nonempty, and for all w having $|N_{D'/2}^{\text{in}}(w)| \geq n^\epsilon$, $S \cap Y_w$ is nonempty.

Run BFS into and out of every $s \in S$. We may assume that $d_{\min}(s, x) \leq D'$ for all $s \in S, x \in V$.

We will construct matrices A and B with rows and columns indexed by vertices in V , as follows: For each vertex $t \in X_u$, let $A[u, t] = 1$. For each vertex $t \in Y_w$, let $B[t, w] = 1$. Multiply A and B using the sparse matrix multiplication algorithm of Theorem 14.

Now, we will consider any pair of vertices (u, w) where u is to the left of w , $u \in R(N_{D'/2}^{\text{in}}(w) \cap S)$, and $w \in L(N_{D'/2}^{\text{out}}(u) \cap S)$. We have that $(A \cdot B)[u, w] > 0$ if and only if $d(u, w) \leq D'$. Indeed, if $d(u, w) \leq D'$, then there is some intermediate vertex t such that $d(u, t) \leq D'/2$ and $d(t, w) \leq D'/2$. Suppose that $t \notin X_u$. Then since X_u is defined as the left-most n^ϵ vertices in $N_{D'/2}^{\text{out}}(u)$, this implies that $|N_{D'/2}^{\text{out}}(u)| > n^\epsilon$ and hence that $|X_u| = n^\epsilon$. Then there is some $s \in S \cap X_u$. Since $t \notin X_u$, t is to the right of all vertices in X_u , and in particular t is to the right of s . This implies $t \notin L(N_{D'/2}^{\text{out}}(u) \cap S)$. But since $w \in L(N_{D'/2}^{\text{out}}(u) \cap S)$ and t lies between u and w , this is a contradiction. Thus, t must be in X_u , and by symmetry, t is in Y_w . So $A[u, t] = 1$ and $B[t, w] = 1$, meaning $(A \cdot B)[u, w] > 0$. Likewise, if $(A \cdot B)[u, w] > 0$, then there exists $t \in X_u \cap Y_w$ such that $d(u, t) \leq D'/2$ and $d(t, w) \leq D'/2$, so $d(u, w) \leq D'$. Therefore, we will mark all pairs (u, w) such that $(A \cdot B)[u, w] > 0$.

Now, consider any $u \in V$ and any $w \notin L(N_{D'/2}^{\text{out}}(u) \cap S)$ to the right of u . We mark the pair (u, w) . If such a w exists, then there is some $s \in N_{D'/2}^{\text{out}}(u) \cap S$ such that s is to the left of or is equal to w . By assumption, $d(s, w) \leq D'$, so $d(u, w) \leq d(u, s) + d(s, w) \leq D'/2 + D' = \frac{3D'}{2}$. By a symmetric argument, for any $w \in V$ and any $u \notin R(N_{D'/2}^{\text{in}}(w) \cap S)$ to the left of w , we have that $d(u, w) \leq \frac{3D'}{2}$, so again we mark any such pair (u, w) . Thus, since we have assumed that $\epsilon(s) \leq \frac{3D'}{2}$ for all $s \in S$, the algorithm will mark all pairs of vertices $u, w \in V$ except those for which we have simultaneously that $u \in R(N_{D'/2}^{\text{in}}(w) \cap S)$ and $w \in L(N_{D'/2}^{\text{out}}(u) \cap S)$.

Finally, check whether there exists an unmarked pair (u, w) . If so, report that $D > D'$. Otherwise, report that $D \leq \frac{3D'}{2}$.

3.2 Runtime Analysis

Here we analyze the runtime of the algorithm of Proposition 15.

► **Lemma 18.** *The algorithm of Proposition 15 runs in time $\tilde{O}(m^{\frac{2\beta}{3\beta+1}} n^{\frac{4\beta+2-\alpha\beta}{3\beta+1}+o(1)} + n^{2+o(1)})$.*

Proof. Topologically sorting the graph takes $O(m \log n)$ time which is absorbed into the final runtime. Constructing for each vertex topologically ordered lists of its in-neighbors and out-neighbors can be done in time $\tilde{O}(n^2)$.

Computing the covering set S takes time $\tilde{O}(n^{1+\epsilon})$ and running BFS from its vertices takes time $O(n^{1-\epsilon} m \log n)$. Checking for each pair (u, w) whether $u \in L(N_{D'/2}^{\text{out}}(u) \cap S)$ and $w \in L(N_{D'/2}^{\text{out}}(u) \cap S)$ can be done in $\tilde{O}(n^2)$ time.

For a fixed u , to compute X_u , we maintain a list of the at most n^ϵ left-most vertices we have found that are at distance $< D'/2$ from u . For each vertex, we store its left-most out-neighbor that is not yet in our set. At each step, we find the left-most such out-neighbor of any vertex in the list; this takes time $O(n^\epsilon)$, and updating the list to reflect that this out-neighbor has been added to our set takes time $O(n^\epsilon)$. At each step we add a vertex to our set X_u , so there are at most $O(n^\epsilon)$ steps. Hence, constructing X_u for a fixed u takes $O(n^{2\epsilon})$ time. Then constructing all sets X_u, Y_w takes $O(n^{1+2\epsilon})$ time altogether.

Finally, note that there are at most n^ϵ 1s in each row of A , since we only set $A[u, t] = 1$ if $t \in X_u$. Thus, A contains at most $n^{1+\epsilon}$ 1s. By symmetry, the same holds for B . Then multiplying A and B can be done in time $O(n^{(1+\epsilon)\frac{2\beta}{\beta+1} + \frac{2-\alpha\beta}{\beta+1} + o(1)} + n^{2+o(1)})$, using Yuster and Zwick's fast sparse matrix multiplication (Theorem 14).

Then the total runtime is:

$$\tilde{O}(n^{1-\epsilon} m + n^{1+2\epsilon} + n^{(1+\epsilon)\frac{2\beta}{\beta+1} + \frac{2-\alpha\beta}{\beta+1} + o(1)} + n^{2+o(1)})$$

Let γ be the largest value such that $n^\gamma = O(m)$. Let $\epsilon = \frac{\alpha\beta + (\beta+1)(\gamma-1)}{3\beta+1}$; this value is chosen because it sets the first and third terms in the above runtime equal (up to $n^{o(1)}$ factors), hence asymptotically minimizing their sum. Substituting the value of ϵ and simplifying, the runtime of the algorithm is:

$$\tilde{O}(n^{\frac{2\beta}{3\beta+1}\gamma + \frac{4\beta+2-\alpha\beta}{3\beta+1} + o(1)} + n^{\frac{2\beta+2}{3\beta+1}\gamma + \frac{\beta-1+2\alpha\beta}{3\beta+1}} + n^{2+o(1)})$$

We note that $3\beta - 3\alpha\beta > 3(\omega - 2) \geq 0 > -1$, giving:

$$4\beta + 2 - \alpha\beta > 2 + (\beta - 1 + 2\alpha\beta) \geq 2\gamma + (\beta - 1 + 2\alpha\beta)$$

Thus, the first term of the above runtime dominates the second. Substituting $n^\gamma = O(m)$, and noting that the polylogarithmic factors in the runtime are of order $n^{o(1)}$, the runtime is $O(m^{\frac{2\beta}{3\beta+1}} n^{\frac{4\beta+2-\alpha\beta}{3\beta+1} + o(1)} + n^{2+o(1)})$, as desired. ◀

References

- 1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant's parser. *SIAM Journal on Computing*, 47(6):2527–2555, 2018.
- 2 Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1681–1697, 2015.
- 3 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. 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, Arlington, VA, USA, January 10-12, 2016*, pages 377–391, 2016.
- 4 Udit Agarwal and Vijaya Ramachandran. Fine-grained complexity for sparse graphs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 239–252, 2018.
- 5 D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999.
- 6 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, 2020.
- 7 B. Ben-Moshe, B. K. Bhattacharya, Q. Shi, and A. Tamir. Efficient algorithms for center problems in cactus networks. *Theoretical Computer Science*, 378(3):237–252, 2007.
- 8 P. Berman and S. P. Kasiviswanathan. Faster approximation of distances in graphs. In *Proc. WADS*, pages 541–552, 2007.
- 9 Michele Borassi, Pierluigi Crescenzi, Michel Habib, Walter A. Kosters, Andrea Marino, and Frank W. Takes. Fast diameter and radius BFS-based computation in (weakly connected) real-world graphs: With an application to the six degrees of separation games. *Theoretical Computer Science*, 586:59–80, 2015.
- 10 Karl Bringmann and Philip Wellnitz. Clique-based lower bounds for parsing tree-adjoining grammars. *arXiv preprint*, 2018. [arXiv:1803.00804](https://arxiv.org/abs/1803.00804).
- 11 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *International Workshop on Parameterized and Exact Computation*, pages 75–85. Springer, 2009.
- 12 T. M. Chan. All-pairs shortest paths for unweighted undirected graphs in $o(mn)$ time. *ACM Transactions on Algorithms*, 8(4):34, 2012.
- 13 Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert Endre Tarjan, and Virginia Vassilevska Williams. Better approximation algorithms for the graph diameter. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1041–1052, 2014.
- 14 V. Chepoi, F. Dragan, and Y. Vaxès. Center and diameter problems in plane triangulations and quadrangulations. In *Proc. SODA*, pages 346–355, 2002.
- 15 V. Chepoi and F. F. Dragan. A linear-time algorithm for finding a central vertex of a chordal graph. In *ESA*, pages 159–170, 1994.
- 16 F. R. K. Chung. Diameters of graphs: Old problems and new results. *Congr. Numer.*, 60:295–317, 1987.
- 17 D.G. Corneil, F.F. Dragan, M. Habib, and C. Paul. Diameter determination on restricted graph families. *Discr. Appl. Math.*, 113:143–166, 2001.
- 18 L. Cowen and C. Wagner. Compact roundtrip routing for digraphs. In *SODA*, pages 885–886, 1999.
- 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)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

- 20 D. Dvir and G. Handler. The absolute center of a network. *Networks*, 43:109–118, 2004.
- 21 D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms and Applications*, 3(3):1–27, 1999.
- 22 Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1150–1162. SIAM, 2012.
- 23 François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1029–1046. SIAM, 2018.
- 24 S.L. Hakimi. Optimum location of switching centers and absolute centers and medians of a graph. *Oper. Res.*, 12:450–459, 1964.
- 25 R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.
- 26 Seth Pettie. A faster all-pairs shortest path algorithm for real-weighted sparse graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 85–97. Springer, 2002.
- 27 Seth Pettie and Vijaya Ramachandran. Computing shortest paths with comparisons and additions. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 267–276, 2002.
- 28 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*, pages 515–524, 2013.
- 29 Virginia Vassilevska Williams, Nike Sun, and Nishith Khandwala. Lecture notes in graph algorithms (hitting sets, APSP), October 2016. URL: <http://theory.stanford.edu/~virgi/cs267/lecture5.pdf>.
- 30 O. Weimann and R. Yuster. Approximating the diameter of planar graphs in near linear time. In *Proc. ICALP*, 2013.
- 31 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.
- 32 Virginia Vassilevska Williams and R Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *Journal of the ACM (JACM)*, 65(5):1–38, 2018.
- 33 C. Wulff-Nilsen. Wiener index, diameter, and stretch factor of a weighted planar graph in subquadratic time. *Technical report, University of Copenhagen*, 2008.
- 34 Raphael Yuster. Computing the diameter polynomially faster than APSP. *arXiv preprint*, 2010. [arXiv:1011.6181](https://arxiv.org/abs/1011.6181).
- 35 Raphael Yuster and Uri Zwick. Fast sparse matrix multiplication. *ACM Trans. Algorithms*, 1(1):2–13, 2005. doi:10.1145/1077464.1077466.
- 36 U. Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM*, 49(3):289–317, 2002.