Fast Approximation Algorithms for the Diameter and Radius of Sparse Graphs

Liam Roditty*
Bar Ilan University
liam.roditty@biu.ac.il

Virginia Vassilevska Williams†
UC Berkeley and Stanford University
virgi@eecs.berkeley.edu

ABSTRACT

The diameter and the radius of a graph are fundamental topological parameters that have many important practical applications in real world networks. The fastest combinatorial algorithm for both parameters works by solving the all-pairs shortest paths problem (APSP) and has a running time of $\widetilde{O}(mn)$ in m-edge, n-node graphs. In a seminal paper, Aingworth, Chekuri, Indyk and Motwani [SODA'96 and SICOMP'99] presented an algorithm that computes in $\widetilde{O}(m\sqrt{n}+n^2)$ time an estimate $\hat{D}$ for the diameter D, such that $\lfloor 2/3D \rfloor \leq \hat{D} \leq D$. Their paper spawned a long line of research on approximate APSP. For the specific problem of diameter approximation, however, no improvement has been achieved in over 15 years.

Our paper presents the first improvement over the diameter approximation algorithm of Aingworth *et al.* , producing an algorithm with the same estimate but with an expected running time of $\widetilde{O}(m\sqrt{n})$. We thus show that for all sparse enough graphs, the diameter can be 3/2-approximated in $o(n^2)$ time. Our algorithm is obtained using a surprisingly simple method of neighborhood depth estimation that is strong enough to also approximate, in the same running time, the radius and more generally, all of the eccentricities, i.e. for every node the distance to its furthest node.

We also provide strong evidence that our diameter approximation result may be hard to improve. We show that if for some constant $\varepsilon > 0$ there is an $O(m^{2-\varepsilon})$ time $(3/2-\varepsilon)$-approximation algorithm for the diameter of undirected unweighted graphs, then there is an $O^*((2-\delta)^n)$ time algorithm for CNF-SAT on n variables for constant $\delta > 0$, and the strong exponential time hypothesis of [Impagliazzo, Paturi, Zane JCSS'01] is false.

*Work supported by the Israel Science Foundation (grant no. 822/10).

†Partially supported by NSF Grants CCF-0830797 and CCF-1118083 at UC Berkeley, and by NSF Grants IIS-0963478 and IIS-0904325, and an AFOSR MURI Grant, at Stanford University.

STOC'13, June 1 - 4, 2013, Palo Alto, California, USA.

Motivated by this negative result, we give several improved diameter approximation algorithms for special cases. We show for instance that for unweighted graphs of *constant* diameter D not divisible by 3, there is an $O(m^{2-\varepsilon})$ time algorithm that gives a $(3/2-\varepsilon)$ approximation for constant $\varepsilon > 0$. This is interesting since the diameter approximation problem is hardest to solve for small D.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]; G.2.2 [**Graph Theory**]: Graph algorithms

Keywords

graph diameter; approximation algorithm; shortest paths

1. INTRODUCTION

The diameter and the radius are two of the most basic graph parameters. The diameter of a graph is the largest distance between its vertices. The center of a graph is a vertex that minimizes the maximum distance to all other nodes, and the radius is the distance from the center to the node furthest from it. Being able to compute the diameter, center and radius of a graph efficiently has become an increasingly important problem in the analysis of large networks [35]. The diameter of the web graph for instance is the largest number of clicks necessary to get from one document to another, and Albert *et al.* were able to show experimentally that it is roughly 19 [2]. The problem of computing a center vertex and the radius of a graph is often studied as a facility location problem for networks: pick a single vertex facility so that the maximum distance from a demand point (client) in the network is minimized.

The algorithmic complexity of the diameter and radius problems is very well-studied. For special classes of graphs there are efficient algorithms [21, 19, 15, 11, 12, 5]. E.g. the radius in chordal graphs can be found in linear time. However, for general graphs with arbitrary edge weights, the *only* known algorithms computing the diameter and radius exactly compute the distance between *every* pair of vertices in the graph, thus solving the all-pairs shortest paths problem (APSP).

For dense directed *unweighted* graphs, one can compute both the diameter and the radius using fast matrix multiplication (this is folklore; for a recent simple algorithm see [17]), thus obtaining $\tilde{O}(n^\omega)$ time algorithms, where $\omega < 2.38$ is the matrix multiplication exponent [14, 33, 34] and n is the number of nodes in the graph. It is not known whether APSP

in such graphs can be solved in $\tilde{O}(n^\omega)$ time – the best algorithm is by Zwick [36] running in $O(n^{2.54})$ time [25], and hence for directed unweighted graphs diameter and radius can be solved somewhat faster than APSP. For undirected unweighted graphs the best known algorithm for diameter and radius is Seidel's $\tilde{O}(n^\omega)$ time APSP algorithm [32].

For *sparse* directed or undirected unweighted graphs, the best known algorithm (ignoring poly-logarithmic factors)[1] for APSP, diameter and radius, does breadth-first search (BFS) from every node and hence runs in $O(mn)$ time, where m is the number of edges in the graph. For sparse graphs with $m = O(n)$, the running time is $\Theta(n^2)$ which is natural for APSP since the algorithm needs to output n^2 distances. However, for the diameter and the radius the output is a single integer, and it is not immediately clear why one should spend $\Omega(n^2)$ time to compute them.

A natural question is whether one can get substantially faster diameter and radius algorithms by settling for an approximation. It is well-known that a 2-approximation for both the diameter and the radius in an undirected graph is easy to achieve in $O(m + n)$ time using BFS from an arbitrary node. On the other hand, for APSP, Dor *et al.* [18] show that any $(2 - \varepsilon)$-approximation algorithm in unweighted undirected graphs running in $T(n)$ time would imply an $O(T(n))$ time algorithm for Boolean matrix multiplication (BMM). Hence apriori it could be that $(2 - \varepsilon)$-approximating the diameter and radius of a graph may also require solving BMM.

In a seminal paper from 1996, Aingworth *et al.* [1] showed that it is in fact possible to get a subcubic $(2 - \varepsilon)$ - approximation algorithm for the diameter in both directed and undirected graphs without resorting to fast matrix multiplication. They designed an $\tilde{O}(m\sqrt{n} + n^2)$ time algorithm computing an estimate $\hat{D}$ that satisfies $\lfloor 2D/3 \rfloor \leq \hat{D} \leq D$. Their algorithm has several important and interesting properties. It is the only known algorithm for approximating the diameter polynomially faster than $O(mn)$ for every m that is superlinear in n. It always runs in truly subcubic time even in dense graphs, and does not explicitly compute all-pairs approximate shortest paths.

For the radius problem, Berman and Kasiviswanathan [6] showed that the approach of Aingworth *et al.* can be used to obtain in $\tilde{O}(m\sqrt{n} + n^2)$ time an estimate $\hat{r}$ that satisfies $r \leq \hat{r} \leq 3/2r$, where r is the radius of the graph. Thus both radius and diameter admit $\tilde{O}(m\sqrt{n} + n^2)$ time 3/2-approximations.

Aingworth *et al.* also presented an algorithm that computes an additive 2-approximation for the APSP problem in $\widetilde{O}(n^{2.5})$ time, that is for every $u, v \in V$ the algorithm returns a value $\hat{d}(u, v)$ such that $d(u, v) \leq \hat{d}(u, v) \leq d(u, v)+2$, where $d(u, v)$ is the distance between u and v. Their paper spawned a long line of research on distance approximation. However, none of the following works considered the specific problems of diameter and radius approximation, but rather focused on approximation algorithms for APSP. Dor, Helperin, and Zwick [18] presented an additive 2-approximation for APSP in unweighted undirected graphs with a running time of $\widetilde{O}(\min\{n^{3/2}m^{1/2}, n^{7/3}\})$, thus improving on Aingworth *et al.* 's APSP approximation algorithm. Baswana *et al.* [3] presented an algorithm for unweighted undirected graphs with an expected running time of $O(m^{2/3}n \log n + n^2)$ that computes an approximation of all distances with a multiplicative error of 2 and an additive error of 1. Elkin [20] presented an algorithm for unweighted undirected graphs with a running time of $O(mn^\rho + n^2\zeta)$ that approximates the distances with a multiplicative error of $(1 + \varepsilon)$ and an additive error that is a function of ζ, ρ and ε. Cohen and Zwick [13] extended the results of [18] to weighted graphs. Baswana and Kavitha [4] presented an $\widetilde{O}(m\sqrt{n}+n^2)$ time multiplicative 2-approximation algorithm and an $\widetilde{O}(m^{2/3}n+n^2)$ time 7/3-approximation algorithm for APSP in weighted undirected graphs.

Since Aingworth *et al.* 's paper, the only paper that considers the diameter approximation problem directly is by Boitmanis *et al.* [9]. They presented an algorithm with $\widetilde{O}(m\sqrt{n})$ running time that computes the diameter with an additive error of $\sqrt{n}$. Although such an additive error could be small for graphs with large diameter, it is prohibitive when it comes to graphs with small diameter.

A simple random sampling argument shows that for all graphs with diameter at least n^δ, there is an $\widetilde{O}(mn^{1-\delta}/\varepsilon)$ time $(1 + \varepsilon)$-approximation algorithm for all $\varepsilon > 0$. Hence diameter approximation is hardest for graphs with small diameter. For such graphs the additive approximation of Boitmanis *et al.* presents no significant approximation guarantee.

Our contributions.

We give the first improvement over the diameter approximation algorithm of Aingworth *et al.* for sparse graphs. We present an algorithm with a slightly better approximation and an expected running time of $\widetilde{O}(m\sqrt{n})$. This is always faster than runtime of [1] for $m = o(n^{1.5})$.

THEOREM 1. *Let $G = (V, E)$ be a directed or an undirected unweighted graph with diameter $D = 3h + z$, where $h \geq 0$ and $z \in \{0, 1, 2\}$. In $\widetilde{O}(m\sqrt{n})$ expected time one can compute an estimate $\hat{D}$ of D such that $2h + z \leq \hat{D} \leq D$ for $z \in \{0, 1\}$ and $2h + 1 \leq \hat{D} \leq D$ for $z = 2$.*

We obtain our efficient algorithm by a surprisingly simple node sampling technique that allows us to replace an expensive neighborhood computation with a cheap estimate.

The diameter and radius are the maximum and minimum *eccentricities* in the graph, respectively. In an unweighted graph, the eccentricity of a vertex is the distance to its furthest node. Our techniques are general enough that we can obtain good estimates of all n eccentricities in an undirected unweighted graph in $\widetilde{O}(m\sqrt{n})$ time. We prove:

THEOREM 2. *Let $G = (V, E)$ be an undirected unweighted graph with diameter D and radius r. In $\widetilde{O}(m\sqrt{n})$ expected time one can compute for every node $v \in V$ an estimate $\hat{e}(v)$ of its eccentricity $ecc(v)$ such that:*

$$\max\{r, 2/3ecc(v)\} \leq \hat{e}(v) \leq \min\{D, 3/2ecc(v)\}.$$

We note that until now the only known approximation algorithm for all node eccentricities that runs in $o(n^2)$ time for sparse graphs is the simple 2-approximation algorithm for radius and diameter that runs BFS from a single node. That algorithm only achieves estimates $\hat{e}(v)$ for which

$$\max\{r, ecc(v)/2\} \leq \hat{e}(v) \leq \min\{D, 2ecc(v)\}.$$

[1] Chan [10] and Blelloch *et al.* [8] presented algorithms with $O(mn/\text{poly}\log n)$ running times.

Our approximation algorithm for radius follows directly from Theorem 2 by taking $\hat{r} = \min_v \hat{e}(v)$. We obtain:

THEOREM 3. *In $\widetilde{O}(m\sqrt{n})$ expected time one can compute an estimate $\hat{r}$ of the radius r of an undirected unweighted graph such that $r \leq \hat{r} \leq 3/2r$.*

Our diameter, radius and eccentricity algorithms naturally extend to graphs with nonnegative edge weights, similar to the algorithm of Aingworth *et al.*

A natural question is whether there is an almost linear time approximation scheme for the diameter problem: an algorithm that for any constant $\varepsilon > 0$ runs in $\tilde{O}(m)$ time and returns an estimate $\hat{D}$ such that $(1-\varepsilon)D \leq \hat{D} \leq D$. Bernstein [7] showed that related problems in directed graphs such as the second shortest path between two nodes and the replacement paths problem admit such approximation schemes. Such an algorithm for diameter would be of immense interest, and has not so far been explicitly ruled out, even conditionally.

Here we give strong evidence that a fast $(3/2-\varepsilon)$ - diameter approximation algorithm may be very hard to find, even for undirected unweighted graphs. We prove:

THEOREM 4. *Suppose there is a constant $\varepsilon > 0$ so that there is a $(3/2-\varepsilon)$-approximation algorithm for the diameter in m-edge undirected unweighted graphs that runs in $O(m^{2-\varepsilon})$ time for every m. Then, SAT for CNF formulas on n variables can be solved in $O^*((2-\delta)^n)$ time for some constant $\delta > 0$.*

The fastest known algorithm for CNF-SAT is the exhaustive search algorithm that runs in $O^*(2^n)$ time by trying all possible 2^n assignments to the variables. It is a major open problem whether there is a faster algorithm. Several other NP-hard problems are known to be equivalent to CNF-SAT so that if one of these problems has a faster algorithm than exhaustive search, then all of them do [16]. Hence, our result has the following surprising implication: if the diameter can be approximated fast enough, then problems such as Hitting Set, Set Splitting, or NAE-SAT, all seemingly unrelated to the diameter, can be solved faster than exhaustive search.

The strong exponential time hypothesis (SETH) of Impagliazzo, Paturi, and Zane [23, 24] implies that there is no improved $O^*((2-\delta)^n)$ time algorithm for CNF-SAT, and hence our result also implies that there is no $(3/2-\varepsilon)$-approximation algorithm for the diameter approximation running in $O(m^{2-\varepsilon})$ time unless SETH fails. (We elaborate on this hypothesis later on in the paper.)

We prove Theorem 4 by showing that any $O(n^{2-\varepsilon})$ time algorithm that distinguishes whether the diameter of a given sparse ($m = O(n)$) undirected unweighted graph is 2 or at least 3 would imply an improved CNF-SAT algorithm. This implies that unless SETH fails, $O(n^2)$ time is essentially required to get a $(3/2-\varepsilon)$-approximation algorithm for the diameter in sparse graphs, within $n^{o(1)}$ factors. Hence, within $n^{o(1)}$ factors, the time for $(3/2-\varepsilon)$-approximating the diameter in a sparse graph is the same as the time required for computing APSP exactly!

In their paper, Aingworth *et al.* showed that one can distinguish between graphs of diameter 2 and 4 in $\widetilde{O}(m\sqrt{n})$ time, whereas we show that distinguishing between 2 and 3 fast may be difficult. We further explore which graph diameters can be efficiently distinguished, and prove the following two theorems that improve upon the approximation of Aingworth *et al.* algorithm.

THEOREM 5. *Let $G = (V, E)$ be a directed or undirected unweighted graph with diameter $D = 3h + z$, where $h \geq 0$ and $z \in \{0, 1, 2\}$. There is an $\tilde{O}(m^{2/3}n^{4/3})$ time algorithm that reports an estimate $\hat{D}$ such that $2h + z \leq \hat{D} \leq D$.*

THEOREM 6. *There is an $\tilde{O}(m^{2/3}n^{4/3})$ time algorithm that when run on an undirected unweighted graph with diameter D, reports an estimate $\hat{D}$ with $\lfloor 4D/5 \rfloor \leq \hat{D} \leq D$.*

Theorem 5 shows for instance that one can efficiently distinguish between directed or undirected graphs of diameter 3 and 5, and Theorem 6 obtains a 5/4-approximation for the diameter that runs in $O(mn/n^{\varepsilon})$ time for some constant $\varepsilon > 0$ in all undirected graphs with a superlinear number of edges. The previous best approximation quality achievable polynomially faster than $O(mn)$ time for such graphs was Aingworth *et al.* 's 3/2-approximation.

We further investigate whether one can ever obtain a $(3/2-\varepsilon)$-approximation for the diameter in $O(m^{2-\varepsilon})$ time, and show that this is indeed possible for graphs with constant diameter that is not divisible by 3. This is intriguing since, as we pointed out earlier, the diameter approximation problem is hardest for graphs with small diameter. We prove:

THEOREM 7. *There is an $\tilde{O}(m^{2-1/(2h+3)})$ time deterministic algorithm that computes an estimate $\hat{D}$ with $\lceil 2D/3 \rceil \leq \hat{D} \leq D$ for all m-edge unweighted graphs of diameter $D = 3h+z$ with $h \geq 0$ and $z \in \{0, 1, 2\}$. In particular, $\hat{D} \geq 2h+z$.*

Notation.

Let $G = (V, E)$ denote a graph. It can be directed or undirected; this will be specified in each context. If the graph is weighted, then there is a function on the edges $w : E \to \mathbb{Q}^+ \cup \{0\}$. Unless explicitly specified, the graphs we consider are unweighted.

For any $u, v \in V$, let $d(u, v)$ denote the distance from u to v in G. Let $BFS^{\text{in}}(v)$ and $BFS^{\text{out}}(v)$ be the incoming and outgoing breadth-first search (BFS) trees of v, respectively, that is the BFS trees starting at v in G and in G with the edges reversed. Let $d^{\text{in}}(v)$ be the depth of $BFS^{\text{in}}(v)$, i.e. the largest distance from a vertex of $BFS^{\text{in}}(v)$ to v. Similarly, let $d^{\text{out}}(v)$ be the depth of $BFS^{\text{out}}(v)$.

In an unweighted graph, the eccentricity of a vertex v denoted with $ecc(v)$ is the depth of its BFS tree $BFS(v)$. In a weighted graph, the eccentricity $ecc(v)$ of v is the maximum over all $u \in V$ of $d(v, u)$. The radius of a graph is $r = \min_{v \in V} ecc(v)$, and the diameter is $D = \max_{v \in V} ecc(v)$.

For $h \leq d^{\text{in}}(v)$, let $BFS^{\text{in}}(v, h)$ be the vertices in the first h levels of $BFS^{\text{in}}(v)$. Similarly, for $h \leq d^{\text{out}}(v)$, let $BFS^{\text{out}}(v, h)$ be the vertices in the first h levels of $BFS^{\text{out}}(v)$.

Let $N_s^{\text{in}}(v)$ ($N_s^{\text{out}}(v)$) be the set of the s closest incoming (outgoing) vertices of v, where ties are broken by taking the vertex with the smaller id. We assume throughout the paper that for each v and each $s \leq n$, $|N_s^{\text{in}}(v)| = |N_s^{\text{out}}(v)| = s$, as otherwise the diameter of the graph would be ∞, and this can be checked with two BFS runs from and to an arbitrary node. For undirected graphs $N_s(v) = N_s^{IN}(v) = N_s^{OUT}(v)$.

Let $d_s^{\text{in}}(v)$ be the largest distance from a vertex of $N_s^{\text{in}}(v)$ to v, and $d_s^{\text{out}}(v)$ be the largest distance from v to a vertex of $N_s^{\text{out}}(v)$. Let $d_s^{\text{in}} = \max_{v \in V} d_s^{\text{in}}(v)$ and $d_s^{\text{out}} = \max_{v \in V} d_s^{\text{out}}(v)$.

For a set $S \subseteq V$ and a vertex $v \in V$ we define $p_S(v)$ to be a vertex of S such that $d(v, p_S(v)) \leq d(v, w)$ for every $w \in S$, i.e. the closest vertex of S to v.

For a degree Δ we define $p_\Delta(v)$ to be the closest vertex to v of degree at least Δ, that is, $d(v, p_\Delta(v)) \leq d(v, w)$ for every $w \in V$ of degree at least Δ.

We use the following standard notation for running times. For a function of n, $f(n)$, $\tilde{O}(f(n))$ denotes $O(f(n)\text{poly}\log n)$ and $O^*(f(n))$ denotes $O(f(n)\text{poly}(n))$.

We write *whp* to mean with high probability, i.e. with probability at least $1 - 1/\text{poly}(n)$.

2. DIAMETER

In this section we present the proof of Theorem 1. We first revisit the algorithm of Aingworth *et al.* and tighten its approximation analysis. We then present our new neighborhood estimation approach that is at the basis of our improved algorithm.

2.1 The algorithm of Aingworth *et al.*

The algorithm of Aingworth, Chekuri, Indyk and Motwani [1], computes a (roughly) 3/2-approximation of the diameter of a directed (or undirected) graph in $\widetilde{O}(m\sqrt{n} + n^2)$ time. Let s be a given parameter in $[1, n]$. The algorithm works as follows. First, it computes $N_s^{\text{out}}(v)$ for every $v \in V$. Then, for a vertex w, where $d_s^{\text{out}}(w) = d_s^{\text{out}}$ it computes $BFS^{\text{out}}(w)$ and for every $u \in N_s^{\text{out}}(w)$ it computes $BFS^{\text{in}}(u)$. Next, it computes a set S that hits $N_s^{\text{out}}(v)$ for every $v \in V$ and for every $u \in S$ it computes $BFS^{\text{out}}(u)$. As an estimate, the algorithm returns the depth of the deepest computed BFS tree. The next lemma appears in [1]. We state it for completeness.

LEMMA 1. *The algorithm runtime is* $\widetilde{O}(ns^2 + (n/s + s)m)$.

Aingworth et al. set $s = \sqrt{n}$ and obtain their running time. We note that if one sets $s = m^{1/3}$ instead, one can get a runtime of $\widetilde{O}(m^{2/3}n)$ that is better for sparse graphs; we later show that both of these runtimes can be improved using our new method.

We now analyze the quality of the estimate returned by the algorithm. Aingworth *et al.* [1] proved that this estimate is at least $\lfloor 2D/3 \rfloor$ in graphs with diameter D. Here we present a tighter analysis.

LEMMA 2. *Let $G = (V, E)$ be a directed graph with diameter $D = 3h + z$, where $h \geq 0$ and $z \in \{0, 1, 2\}$. Let $\hat{D}$ be the estimate returned by the algorithm. For $z \in \{0, 1\}$, we have $2h + z \leq \hat{D} \leq D$. For $z = 2$, we have that $2h + 1 \leq \hat{D} \leq D$.*

PROOF. Let $a, b \in V$ such that $d(a, b) = D$. First notice that the algorithm always returns the depth of some shortest paths tree and hence $\hat{D} \leq D$.

If $d_s^{\text{out}}(w) \leq h$ then also $d_s^{\text{out}}(a) \leq h$ and as S hits $N_s^{\text{out}}(a)$, one of the BFS trees computed for vertices of S has depth at least $2h + z$. Hence, assume that $d_s^{\text{out}}(w) > h$. We can also assume that $d^{\text{out}}(w) < 2h + z$ as otherwise when we compute $BFS^{\text{out}}(w)$, the estimate would become at least $2h + z$.

As $d^{\text{out}}(w) < 2h + z$, also $d(w, b) < 2h + z$. Since $d_s^{\text{out}}(w) > h$, we have that $BFS^{\text{out}}(w, h) \subseteq N_s^{\text{out}}(w)$. Hence there is a vertex $w' \in N_s^{\text{out}}(w)$ on the path from w to b such that $d(w, w') = h$ and hence $d(w', b) < h + z$. Since $d(a, b) = 3h + z$, we must have that $d(a, w') \geq 2h + 1$. As the algorithm computes $BFS^{\text{in}}(u)$ for every $u \in N_s^{\text{out}}(w)$, in particular, it computes $BFS^{\text{in}}(w')$, and returns an estimate $\geq 2h + 1$. For $z \in \{0, 1\}$, $d(a, w') \geq 2h + 1 \geq 2h + z$ and hence the final estimate returned is always at least $2h + z$. For $z = 2$ we only have that $d(a, w') \geq 2h + 1$ and if the algorithm returns $d(a, w')$ as an estimate, it may return $2h + 1$ instead of $2h + z$. □

2.2 Improving the running time

The algorithm of Aingworth *et al.* [1] runs in $\widetilde{O}(ns^2 + (n/s + s)m)$. In this section we show how to get rid of the ns^2 term with some randomization, while keeping the quality of the estimate unchanged. By choosing $s = \sqrt{n}$, we get an algorithm running in $\widetilde{O}(m\sqrt{n})$ time.

The term of ns^2 in the running time comes from the computation of $N_s^{\text{out}}(v)$ for every $v \in V$. This computation is done to accomplish two tasks. One task is to obtain $d_s^{\text{out}}(v)$ for every $v \in V$ and then to use it to find a vertex w such that $d_s^{\text{out}}(w) = d_s^{\text{out}}$. A second task is to obtain, deterministically, a hitting set S of size $\widetilde{O}(n/s)$ that hits the set $N_s^{\text{out}}(v)$ of every $v \in V$.

Our main idea is to accomplish these two tasks without explicitly computing $N_s^{\text{out}}(v)$ for every $v \in V$. The major step in our approach is to completely modify the first task above by picking a different type of vertex to play the role of w. Making the second task above fast can be accomplished easily with randomization. We elaborate on this below.

Our algorithm works as follows. First, it computes a hitting set by using randomization, that is, it picks a random sample S of the vertices of size $\Theta(n/s \log n)$. This guarantees that with high probability (at least $1 - n^{-c}$, for some constant c), $S \cap N_s^{\text{out}}(v) \neq \emptyset$, for every $v \in V$. This accomplishes the second task above in $\tilde{O}(n)$ time, with high probability. Similarly to the algorithm of Aingworth *et al.* [1], our algorithm computes $BFS^{\text{out}}(v)$, for every $v \in S$.

We now explain the main idea of our algorithm, i.e. how to replace the first task above with a much faster step. First, for every $v \in V$ our algorithm computes the closest node of S, $p_S(v)$, to v, by creating a new graph as follows. It adds an additional vertex r with edges (u, r), for every $u \in S$. It computes $BFS^{\text{in}}(r)$ in this graph. It is easy to see that for every $v \in V$ the last vertex before r on the shortest path from v to r is $p_S(v)$. This step takes $O(m)$ time.

Now, the crucial point of our algorithm is that, as opposed to the algorithm of Aingworth *et al.* that picks a vertex w such that $d_s^{\text{out}}(w) = d_s^{\text{out}}$, our algorithm finds a vertex $w \in V$ that is furthest away from S: i.e. such that $d(w, p_S(w)) \geq d(u, p_S(u))$, for every $u \in V$. The vertex w plays the same role as its counterpart in [1]: Our algorithm computes $BFS^{\text{out}}(w)$ and obtains $N_s^{\text{out}}(w)$ from it. Finally, it computes $BFS^{\text{in}}(u)$ for every $u \in N_s^{\text{out}}(w)$. As an estimate, the algorithm returns the depth of the deepest BFS tree that it has computed.

In the next Lemma we analyze the running time of the algorithm.

LEMMA 3. *The algorithm runtime is* $\widetilde{O}((n/s+s)m)$.

PROOF. A hitting set S is formed in $\tilde{O}(n)$ time. With a single BFS computation, in $O(m)$ time, we find $p_S(v)$ for every $v \in V$, and hence also find w. The cost of computing a BFS tree for every $v \in S \cup N_s^{\text{out}}(w)$ is $\widetilde{O}((n/s+s)m)$. □

Next, we show that the estimate produced by our algorithm is of the same quality as the estimate produced by Aingworth *et al.* algorithm, whp.

LEMMA 4. *Let $G=(V,E)$ be a directed (or undirected) graph with diameter $D=3h+z$, where $h \geq 0$ and $z \in \{0,1,2\}$. Let $\hat{D}$ be the estimate returned by the above algorithm. With high probability, $2h+z \leq \hat{D} \leq D$ whenever $z \in \{0,1\}$, and $2h+1 \leq \hat{D} \leq D$ whenever $z=2$.*

PROOF. Let $a,b \in V$ such that $d(a,b)=D$. Let w be a vertex that satisfies $d(w,p_S(w)) \geq d(u,p_S(u))$, $\forall u \in V$.

If $d(w,p_S(w)) \leq h$ then also $d(a,p_S(a)) \leq h$. As the algorithm computes $BFS^{\text{out}}(v)$ for every $v \in S$, it follows that $BFS^{\text{out}}(p_S(a))$ is computed as well and its depth is at least $2h+z$ as required. Hence, assume that $d(w,p_S(w)) > h$. We can assume also that $d^{\text{out}}(w) < 2h+z$ since the algorithm computes $BFS^{\text{out}}(w)$ and if $d^{\text{out}}(w) \geq 2h+z$ then it computes a BFS tree of depth at least $2h+z$.

Since $d^{\text{out}}(w) < 2h+z$ it follows that $d(w,b) < 2h+z$. Moreover, since $d(w,p_S(w)) > h$ and S hits $N_s^{\text{out}}(w)$ whp, we must have that $N_s^{\text{out}}(w)$ contains a node at distance $> h$ from w, and hence $BFS^{\text{out}}(w,h) \subseteq N_s^{\text{out}}(w)$. This implies that there is a vertex $w' \in N_s^{\text{out}}(w)$ on the path from w to b such that $d(w,w')=h$ and hence $d(w',b) < h+z$. Since $d(a,b)=3h+z$, we also have that $d(a,w') \geq 2h+1$.

The algorithm computes $BFS^{\text{in}}(u)$ for every $u \in N_s^{\text{out}}(w)$, and in particular, it computes $BFS^{\text{in}}(w')$, thus returning an estimate at least $d(a,w') \geq 2h+1$. Hence for $z \in \{0,1\}$ the final estimate is always $\geq 2h+z$, and for $z=2$ the estimate could be $2h+1$ but no less. □

We now turn to prove Theorem 1 from the introduction.

Reminder of Theorem 1 *Let $G=(V,E)$ be a directed or an undirected graph with diameter $D=3h+z$, where $h \geq 0$ and $z \in \{0,1,2\}$. In $\widetilde{O}(m\sqrt{n})$ expected time one can compute an estimate $\hat{D}$ of D such that $2h+z \leq \hat{D} \leq D$ for $z \in \{0,1\}$ and $2h+1 \leq \hat{D} \leq D$ for $z=2$.*

PROOF. From Lemma 3 we have that if we set $s=\sqrt{n}$ the algorithm runs in $\widetilde{O}(m\sqrt{n})$ worst case time. From Lemma 4 we have that whp, the algorithm returns an estimate of the desired quality. We now show how to convert the algorithm into a Las-Vegas one so that it always returns an estimate of the desired quality but the running time is $\widetilde{O}(m\sqrt{n})$ in expectation.

Randomization is used only in order to obtain a set that hits $N_s^{\text{out}}(v)$ for every $v \in V$. The only place that the hitting set affects the quality of the approximation is in Lemma 4 where we used the fact that, whp, S contains a node of $N_s^{\text{out}}(w)$, so that if $d(w,S) > h$, $N_s^{\text{out}}(w)$ contains a node at distance $> h$ from w.

Algorithm 1: Approx-Ecc(G)

```
Let S be a random sample of Θ(n/s log n) nodes.
Let w be such that d(w, p_S(w)) ≥ d(u, p_S(u)) for all
u ∈ V.
foreach x ∈ N_s(w) ∪ S do
    BFS(x).
foreach v ∈ V do
    if d(v, v_t) ≤ d(v_t, w) then
        ê(v) = max{max_{q∈S} d(v,q), d(v,w), ecc(v_t)}
    else
        ê(v) =
        max{max_{q∈S} d(v,q), d(v,w), min_{q∈S} ecc(s)}
```

Note that the algorithm computes $N_s^{\text{out}}(w)$ and we can check whether S intersects it in $\widetilde{O}(s)$ time. If it does not, we can rerun the algorithm until we have verified that $S \cap N_s^{\text{out}}(w) \neq \emptyset$. In each run, $S \cap N_s^{\text{out}}(w) = \emptyset$ holds with very small probability: S is large enough so that whp it intersects the s-neighborhoods of all n vertices of the graph. Thus, the expected running time of the algorithm is $\widetilde{O}(m\sqrt{n})$ and its estimate is guaranteed to have the required quality. □

Just as in [1], our algorithm works for graphs with nonnegative weights as well by replacing every use of BFS with Dijkstra's algorithm. The proofs are analogous, the running time is increased by at most a $\log n$ factor, and the quality of the approximation only suffers an additive W term, where W is the maximum edge weight in the graph. (The same approximation quality is achieved by Aingworth *et al.* but with an $\widetilde{O}(m\sqrt{n}+n^2)$ running time.) We obtain:

THEOREM 8. *Let $G=(V,E)$ be a directed or an undirected graph with nonnegative edge weights at most W and diameter D. In $\widetilde{O}(m\sqrt{n})$ expected time one can compute an estimate $\hat{D}$ of D such that $\lfloor 2D/3 - W \rfloor < \hat{D} \leq D$.*

3. ECCENTRICITIES

In this section we show that our method can be generalized to compute for every vertex v in an undirected unweighted graph, a good approximation $\hat{e}(v)$ of its eccentricity $ecc(v)$. We prove Theorem 2.

Reminder of Theorem 2 *Let $G=(V,E)$ be an undirected graph with diameter D and radius r. In $\widetilde{O}(m\sqrt{n})$ expected time one can compute for every node $v \in V$ an estimate $\hat{e}(v)$ of its eccentricity $ecc(v)$ such that:*

$$\max\{r, 2/3 ecc(v)\} \leq \hat{e}(v) \leq \min\{D, 3/2 ecc(v)\}.$$

We note that our eccentricities algorithm can also be made to work for undirected graphs with nonnegative weights at most W by again using Dijkstra's algorithm in place of BFS. Then the running time is still $\widetilde{O}(m\sqrt{n})$ and the approximation quality becomes $2/3 ecc(v) - 2W < \hat{e}(v) < 3/2 ecc(v) + W$.

One can immediately obtain our 3/2-approximation of the radius in unweighted undirected graphs stated in Theorem 3 as a corollary to Theorem 2 by taking $\hat{r} = \min_v \hat{e}(v)$. For this choice, $\hat{r} \geq r$, and $\hat{r} \leq \min_v 3/2 ecc(v) = 3/2 r$.

The algorithm starts similarly to the algorithm for diameter. It first picks a random set S on $O(\sqrt{n}\log n)$ nodes, and finds the vertex w furthest from S. Then it computes all BFS trees for the vertices of $S \cup N_s(w)$ for $s = \sqrt{n}$. Let $v_t \in N_s(w)$ be the closest vertex to v on the shortest path between w and v. Such a vertex exists since $w \in N_s(w)$, and for every v it can be computed during the computation of the BFS tree from w.

The main idea in computing estimates for the eccentricities is to compare between $d(v, v_t)$ and $d(v_t, w)$ for each v. Let $e'(v) = \max\{\max_{q\in S} d(v, q), d(v, w)\}$. The algorithm sets $\hat{e}(v)$ as follows:

$$\hat{e}(v) = \begin{cases} \max\{e'(v), ecc(v_t)\} & \text{if } d(v, v_t) \le d(v_t, w) \\ \max\{e'(v), \min_{q\in S} ecc(s)\} & \text{if } d(v, v_t) > d(v_t, w) \end{cases}$$

The algorithm is presented in Algorithm 1. It is straightforward to see that it runs in $\widetilde{O}(m\sqrt{n})$ time when s is set to $\sqrt{n}$. In the next three lemmas we prove the bounds on the approximation.

LEMMA 5. *For every $v \in V$, $\hat{e}(v) \le 3/2 ecc(v)$.*

PROOF. We divide the proof into two cases:

Case 1: $[d(v, v_t) \le d(v_t, w)]$ In this case we only need to show that $ecc(v_t) \le 3/2 ecc(v)$ as $\max_{q\in S} d(v, q) \le ecc(v)$ and $d(v, w) \le ecc(v)$. Since $d(v, v_t) \le d(v_t, w)$, it follows that $d(v, v_t) \le d(v, w)/2$, and hence $d(v, v_t) \le ecc(v)/2$. From the triangle inequality we have $ecc(v_t) \le d(v_t, v) + ecc(v)$, thus, $d(v_t) \le 3/2 ecc(v)$.

Case 2: $[d(v, v_t) > d(v_t, w)]$ We only need to show that $\min_{q\in S} d(q) \le 3/2 ecc(v)$. Since $d(v, v_t) > d(v_t, w)$, we must have $d(v_t, w) < d(v, w)/2 \le ecc(v)/2$.

Now, since S hits the set $N_s(w)$ with high probability, every node at distance $< d(w, S)$ from w is in $N_s(w)$. Consider the node v'_t that is after v_t on the shortest path between w and v. Since v_t is the closest node to v on the shortest path between w and v that belongs to $N_s(w)$ it follows that $v'_t \notin N_s(w)$. Moreover, since $d(w, v'_t) = d(w, v_t)+1$ it follows that $d(w, v'_t) \le ecc(v)/2$, and so if $d(w, S) > ecc(v)/2$, then $v'_t \in N_s(w)$ which would be a contradiction. Hence $d(w, S) \le ecc(v)/2$. But as w is the vertex that is furthest from S, $d(w, S) \ge d(v, S)$ and it follows that $d(v, S) \le ecc(v)/2$. Now if $d(v, q') = d(v, S)$ and $q'' = \arg\min_{q\in S} ecc(q)$, then $ecc(q'') \le ecc(q') \le d(q', v) + ecc(v) \le 3/2 ecc(v)$. □

LEMMA 6. *For every $v \in V$, $\hat{e}(v) \ge 2/3 ecc(v)$.*

PROOF. If $\max_{q\in S} d(v, q) \ge 2/3 ecc(v)$ then we are done since our estimate is always at least as large as this. Hence assume that for all $q \in S$, $d(v, q) < 2/3 ecc(v)$. Let x_v be the other endpoint of the eccentricity path from v. Then, $d(S, x_v) > ecc(v)/3$ since $ecc(v) \le d(v, q)+d(q, x_v) < 2/3 ecc(v) + d(q, x_v)$ for all $q \in S$. Since w is the furthest node from S, we must also have $d(w, S) > ecc(v)/3$. Since S hits $N_s(w)$ with high probability, all nodes at distance $\le ecc(v)/3$ from w must be in $N_s(w)$. Hence, $d(w, v_t) \ge ecc(v)/3$.

Now we have two cases:

Case 1: $[d(v, v_t) > d(v_t, w)]$ Here we return an estimate that is at least $d(v, w) = d(v, v_t) + d(v_t, w) > 2d(v_t, w) \ge 2/3 ecc(v)$.

Case 2: $[d(v, v_t) \le d(v_t, w)]$ Here $d(v, v_t) = d(v, w) - d(v_t, w) \le d(v, w) - ecc(v)/3$. Since we are done if $d(v, w) \ge 2/3 ecc(v)$, assume that $d(v, w) < 2/3 ecc(v)$, and so $d(v, v_t) \le ecc(v)/3$. By the triangle inequality, $ecc(v_t) \ge ecc(v) - d(v, v_t) \ge 2/3 ecc(v)$. □

LEMMA 7. *For every $v \in V$, $\hat{e}(v) \in [r, D]$.*

PROOF. In all cases, we return a distance in the graph, so that $\hat{e}(v) \le D$. Moreover, our algorithm works in such a way that for every $v \in V$ there exists a vertex $v' \in V$ such that $\hat{e}(v) \ge ecc(v')$, hence, $\hat{e}(v) \ge r$. □

4. HARDNESS UNDER SETH

Impagliazzo, Paturi, and Zane [23, 24] introduced the Exponential Time Hypothesis (ETH) and its stronger variant, the Strong Exponential Time Hypothesis (SETH). These two complexity hypotheses assume lower bounds on how fast satisfiability problems can be solved. They have frequently been used as a basis for conditional lower bounds for other concrete computational problems. ETH states that 3-SAT on n variables and m clauses cannot be solved in $2^{\delta n}\text{poly}(m, n)$ time for some $\delta > 0$.

A natural question is how fast can one solve r-SAT as r grows. Impagliazzo, Paturi, and Zane define:

$$s_r = \inf\{\delta \mid \exists O^*(2^{\delta n}) \text{ time algorithm solving } r\text{-SAT instances with } n \text{ variables}\},$$

and $s_\infty = \lim_{r\to\infty} s_r$.

The sequence s_r is clearly nondecreasing. Impagliazzo, Paturi, and Zane show that if ETH holds, then s_r also increases infinitely often. Furthermore, all known algorithms for r-SAT nowadays take time $O(2^{n(1-c/r)})$ for some constant c independent of n and r (e.g. [22, 26, 28, 27, 30, 31]). Because of this, it seems plausible that $s_\infty = 1$, and this is exactly the strong exponential time hypothesis.

HYPOTHESIS 1 ([23, 24]). **SETH***: $s_\infty = 1$.*

One immediate consequence of SETH is that CNF-SAT on n variables cannot be solved in $2^{n(1-\varepsilon)}\text{poly}(n)$ time for any $\varepsilon > 0$. The best known algorithm for CNF-SAT is the $O^*(2^n)$ time exhaustive search algorithm which tries all possible 2^n assignments to the variables, and it has been a major open problem to obtain an improvement. Cygan et al. [16] showed that SETH is also equivalent to the assumption that several other NP-hard problems cannot be solved faster than by exhaustive search, and the best algorithms for these problems are the exhaustive search ones.

Assuming SETH, one can prove tight conditional lower bounds on the complexity of some problems in P as well. Pătraşcu and Williams [29] give several tight lower bounds (matching the known upper bounds) for problems such as k-dominating set (for any constant $k \ge 3$), 2SAT with two extra unrestricted length clauses, and HornSAT with k extra unrestricted length clauses.

For constant k, k-dominating set is defined as follows: given an undirected graph $G = (V, E)$, is there a set S of k vertices so that every vertex $v \in V$ is either in S or has an edge to some vertex in S?

The best algorithm for k-dominating set for $k \ge 7$ runs in $n^{k+o(1)}$ time, and obtaining $O(n^{k-\varepsilon})$ time would break SETH [29]. The k-dominating set problem is well-studied in the area of fixed-parameter complexity. It is complete for W[2], and improving on the $n^{k+o(1)}$ running time is a major open problem. In this section we will prove that fast diameter approximation in sparse graphs would not only falsify SETH, but that it would imply faster algorithms for k-

dominating set as well, a problem that could be potentially harder than CNF-SAT. [2]

THEOREM 9. *Suppose one can distinguish between diameter 2 and 3 in an m-edge undirected unweighted graph in time $O(m^{2-\varepsilon})$ for some constant $\varepsilon > 0$. Then for all integers $k \geq 2/\varepsilon$, $2k$-dominating set can be solved in $O^*(n^{2k-\varepsilon})$ time. Moreover, CNF-SAT on n variables and m clauses is in $O(2^{n(1-\varepsilon/2)}poly(m,n))$ time, and SETH is false.*

Remark: Theorem 9 immediately implies Theorem 4 in the introduction, as any $(3/2-\varepsilon)$-approximation algorithm can distinguish between diameter 2 and 3.

PROOF. Given an instance $G = (V, E)$ of $2k$-Dominating set for constant k, we construct an instance of the 2 vs 3 diameter problem and we show that $2k$-Dominating set in n-node graphs can be solved in $O^*(n^{2k-\delta})$ time for some constant $\delta > 0$ depending on ε.

Take all k-subsets of the vertices in V and add a node for each of them to the 2 vs 3 instance G'. Add a node for every vertex in V – call this set of nodes V' and make V' into a clique.

For every k-subset S of vertices of V, connect S to $v \in V'$ in G' iff S *does not dominate* v in G. While we do this we check whether each S is a k-dominating set in G, and if so, we stop. From now on we can assume that none of the k-subsets S are dominating sets in G.

Now, notice that if S and T are two k-subsets so that their union is not a $(\leq 2k)$-dominating set in G, then the distance in G' between S and T is 2: there is some u that is dominated by neither S nor T and so $S - u - T$ is a path of length 2. If, on the other hand, $S \cup T$ is a dominating set in G, then there is no such path and the shortest path between S and T in G' is to go from S to some v that S doesn't dominate, then to some u that T doesn't dominate (V' is a clique) and then from u to T.

The distance between any u and v in V' is 1, and the distance between any u and any S is at most 2: go from u to some node v that S doesn't dominate and then to S.

Hence, if there is no $2k$-dominating set in G, then the diameter of G' is 2, and if there is one, then the diameter of G' is 3. G' has $\binom{n}{k} + n$ nodes and at most $O(n \cdot \binom{n}{k}) \leq O(n^{k+1})$ edges.

Since we can solve the diameter problem in $O(m^{2-\varepsilon})$ time, applying that algorithm to G' solves $2k$-dominating set in G for any $k \geq 2$ in time $O(n^{2k+2-\varepsilon k-\varepsilon})$.

We want this to be $O(n^{2k-\delta})$ for some $\delta > 0$, so it suffices to pick k so that $-\delta \geq 2 - \varepsilon(k+1)$. If we want $\delta = \varepsilon$, then $k \geq 2/\varepsilon$ suffices.

To prove the statement for CNF-SAT, one can apply the reduction from [29], and one would obtain that a $O(n^{2-\varepsilon})$ time algorithm for diameter approximation would imply an $O^*(2^{n(1-\varepsilon^2/4)})$ time algorithm for CNF-SAT. Here we show a direct reduction from CNF-SAT to diameter that gives the runtime given in the theorem.

Given an instance of CNF-SAT on n variables and m clauses, we first partition the variables into two sets S_1 and S_2 on $n/2$ variables each. Create a vertex for every one of the $2^{n/2}$ partial assignments to the variables in S_1 and similarly a vertex for every assignment to the variables in S_2. Create two nodes t_1 and t_2 and add an edge to t_i from each assignment to the variables of S_i. Create a node for every clause, and connect all clause nodes together with t_1 and t_2 into a clique of size $m+2$. Then, similarly to the reduction from k-dominating set, connect every assignment node to the clauses that it does not satisfy. Now, this graph has diameter 3 iff there are two partial assignments, ϕ_1 to S_1 and ϕ_2 to S_2 that together form a satisfying assignment to the CNF formula, i.e. the distance between ϕ_1 and ϕ_2 in the graph is 3 iff they form a satisfying assignment, and all other node distances are ≤ 2. The graph has $O(m + 2^{n/2})$ nodes and $O(m2^{n/2})$ edges. The statement follows. □

[2] Pătraşcu and Williams [29] are able to show that improving the runtime for k-dominating set can be reduced to improving the known algorithms for a problem related to CNF-SAT, but that problem could still be harder than CNF-SAT.

5. IMPROVED APPROXIMATIONS

In this section we show that in some cases it is possible to obtain fast $(3/2-\varepsilon)$-approximations for the diameter. We present two algorithms, one works well for dense graphs and the other for sparse graphs.

5.1 Dense graphs

Here we prove Theorems 5 and 6. Both theorems rely on algorithm Approx-Diam(G) that works as follows. First, it runs the Aingworth *et al.* algorithm both on the input graph G and on the input graph with the edge directions reversed, G^R. Let $\hat{D}$ be the maximum value returned by these two runs. A byproduct of this step is that for every $v \in V$ we have computed $BFS^{\text{out}}(v, d_s^{\text{out}}(v)-1)$ and $BFS^{\text{in}}(v, d_s^{\text{in}}(v)-1)$. Next, the algorithm scans all pairs of vertices u and v and checks whether the following condition holds: $BFS^{\text{out}}(u, d_s^{\text{out}}(u)-1)$ and $BFS^{\text{in}}(v, d_s^{\text{in}}(v)-1)$ are disjoint and there is no edge between $BFS^{\text{out}}(u, d_s^{\text{out}}(u)-1)$ and $BFS^{\text{in}}(v, d_s^{\text{in}}(v)-1)$. Given vertices u and v for which the condition holds, the algorithm updates $\hat{D}$ to be the maximum between its current value and $d_s^{\text{out}}(u) + d_s^{\text{in}}(v)$.

We start by showing that the estimate reported by the algorithm is upper-bounded by the graph diameter.

LEMMA 8. *Let $G = (V, E)$ be a graph of diameter D. If $\hat{D} =$ Approx-Diam(G), then $\hat{D} \leq D$.*

PROOF. If Approx-Diam(G) returns the value that it gets from one of the runs of Aingworth *et al.* algorithm then the claim follows from Lemma 2. If the algorithm reports $d_s^{\text{out}}(u) + d_s^{\text{in}}(v)$ for some pair of vertices $u, v \in V$ it is because there is no edge from $BFS^{\text{out}}(u, d_s^{\text{out}}(u)-1)$ to $BFS^{\text{in}}(v, d_s^{\text{in}}(v)-1)$, and no vertex in common between the two trees. This means that there is no path of length at most $d_s^{\text{out}}(u) + d_s^{\text{in}}(v) - 1$ from u to v, and hence, any path from u to v, and in particular the shortest one, is of length at least $d_s^{\text{out}}(u) + d_s^{\text{in}}(v) \leq D$ as required. □

Next, we lower-bound the diameter estimate $\hat{D}$.

LEMMA 9. *Let $G = (V, E)$ be a graph of diameter $D = 3h + z$, where $h \geq 1$ and $z \in \{0, 1, 2\}$. If $\hat{D} =$ Approx-Diam(G) then $2h + z \leq \hat{D} \leq 3h + z$.*

PROOF. Let $a, b \in V$ such that $d(a, b) = D$. Running the algorithm of Aingworth *et al.* for G and the reverse G^R of G implies that we get an approximation of $2h + z$ in the following cases.

Case 1: $[z \neq 2]$. From Lemma 2, we have that the estimate is at least $2h + z$.

Case 2: $[d_s^{\text{out}}(a) \leq h$ or $d_s^{\text{in}}(b) \leq h]$. If $d_s^{\text{out}}(a) \leq h$ then the hitting set computed by the Aingworth *et al.* algorithm contains a vertex at distance at most h from a and hence one of the BFS trees that it computes has depth at least $2h + z$. Running the algorithm on G^R guarantees that the same holds when $d_s^{\text{in}}(b) \leq h$.

Case 3: $[\exists w \in V$ s.t. $d_s^{\text{out}}(w) \geq h + 2]$. In this case let w be the vertex with the largest $d_s^{\text{out}}(w)$ value. The Aingworth *et al.* algorithm computes $BFS^{\text{out}}(w)$. If $d^{\text{out}}(w) \geq 2h + 2$ then the claim holds so assume that $d^{\text{out}}(w) \leq 2h + 1$. The algorithm computes $BFS^{\text{in}}(v)$ for every $v \in BFS^{\text{out}}(w, h + 1)$ and since $d(w, b) \leq 2h + 1$ there is a vertex $w' \in BFS^{\text{out}}(w, h + 1)$ such that $d(w', b) \leq h$. As the algorithm computes $BFS^{\text{in}}(w')$ and $d(a, w') \geq 2h + z$ the claim holds.

For the rest of the proof we assume that the three cases above do not hold, hence, $z = 2$, $d_s^{\text{out}}(a) = h + 1$ and $d_s^{\text{in}}(b) = h + 1$. The second part of our algorithm searches for a pair of vertices $u, v \in V$ such that there is no edge from $BFS^{\text{out}}(u, d_s^{\text{out}}(u) - 1)$ to $BFS^{\text{in}}(v, d_s^{\text{in}}(v) - 1)$ (and no vertex in common between the two trees). As $D = d(a, b) = 3h + 2 > 2h + 1$, and $d_s^{\text{out}}(a) - 1 = h$ and $d_s^{\text{in}}(b) - 1 = h$, we have that there is no edge from $BFS^{\text{out}}(a, d_s^{\text{out}}(a) - 1)$ to $BFS^{\text{in}}(b, d_s^{\text{in}}(b) - 1)$ (and no vertex in common between the two trees). Since the estimate reported by the algorithm is the maximum among values that also include $d_s^{\text{out}}(a) + d_s^{\text{in}}(b) = 2h + 2$, we get that $\hat{D} \geq 2h + 2$, as required. □

Reminder of Theorem 5 *Let $G = (V, E)$ be a directed or undirected unweighted graph with diameter $D = 3h + z$, where $h \geq 0$ and $z \in \{0, 1, 2\}$. There is an $\tilde{O}(m^{2/3}n^{4/3})$ time algorithm that reports an estimate $\hat{D}$ with $2h+z \leq \hat{D} \leq D$.*

Proof. The bounds on the estimate follow from Lemma 9 and Lemma 8. Running the algorithm of Aingworth *et al.* takes $\widetilde{O}(m(s + n/s) + ns^2)$ time. Finding a pair of vertices $u, v \in V$ such that there is no edge from $BFS^{\text{out}}(u, d_s^{\text{out}}(u) - 1)$ to $BFS^{\text{in}}(v, d_s^{\text{in}}(v) - 1)$ takes $O(n^2s^2)$ time. Setting $s = (m/n)^{1/3}$ gives us the running time. □

We can use Theorem 5 to obtain an even better approximation for undirected graphs.

Reminder of Theorem 6 *There is an $\tilde{O}(m^{2/3}n^{4/3})$ time algorithm that in undirected unweighted graphs with diameter D, reports an estimate $\hat{D}$ with $\lfloor 4D/5 \rfloor \leq \hat{D} \leq D$.*

Proof. Using [18] we compute the distances between every pair of vertices in the graph, with an additive error of 2 in $O(\min(n^{3/2}\sqrt{m}, n^{7/3}))$ time. If $\hat{D}$ is the maximum distance minus 2 then $D-2 \leq \hat{D} \leq D$. For every $D \geq 6$ we have that $D - 2 \geq \lfloor 4/5D \rfloor$. Thus, when $\hat{D} \geq 4$ we get an estimate of at least $\lfloor 4D/5 \rfloor$. If $\hat{D} = 3$ then D might be either 3, 4 or 5, that is, $D = 3 + z$, where $z \in \{0, 1, 2\}$. If $D = 5$, an estimate of 3 is not good enough, thus we run Approx-Diam(G). Let D' be the estimate reported by Approx-Diam(G). From Lemma 9 it follows that if $D = 5$ then $D' \geq 4$ and we have the required approximation. If $\hat{D} = 2$ then D might be either 2, 3 or 4, and for this case we can just use the Aingworth *et al.* algorithm to get an estimate of 3 whenever $D = 4$ which gives the desired approximation. □

5.2 Sparse graphs

We now show that for graphs of constant diameter, it is sometimes possible to obtain a better than 3/2-approximation in $\widetilde{O}(m^{2-\varepsilon})$ time for constant $\varepsilon > 0$.

Our result is based on algorithm Approx-Diam-Sparse($G, \tilde{h}$). This algorithm is given an estimate $\tilde{h}$ of h so that $\tilde{h} \geq h$ and works as follows. Let Δ be a parameter and let H be the set of vertices of outdegree at least Δ. For every vertex of H, the algorithm computes an outgoing BFS tree. Then, it computes the distance from every node in $V \setminus H$ to H. This is done by adding an extra node r to the graph with edges from each node of H to r and then computing an incoming BFS to r in $O(m)$ time. The distance of a node v to H is its distance to r, minus 1. The algorithm then picks the vertex w that is furthest from H and computes $BFS^{\text{out}}(w)$. Let $h' = \min\{\tilde{h} + 1, d(w, H)\}$. The algorithm computes $BFS^{\text{in}}(v)$ for every $v \in BFS^{\text{out}}(w, h')$. Finally, it returns the maximum depth of all computed BFS trees.

We now analyze the quality of the approximation.

Lemma 10. *Let $G = (V, E)$ be a graph of constant diameter $D = 3h + z$, where $h \geq 0$ and $z \in \{0, 1, 2\}$. If $\hat{D} =$ Approx-Diam-Sparse($G, \tilde{h}$) for $\tilde{h} \geq h$, then $2h+z \leq \hat{D} \leq D$.*

Proof. First notice, that in any case the algorithm returns the depth of some BFS tree in the graph, thus $\hat{D} \leq D$.

Now, let $a, b \in V$ such that $d(a, b) = D$ and let $H \subseteq V$ be the set of vertices of outdegree at least Δ. Let $y^o \in H$ be the vertex with the deepest outgoing BFS in H. Let y^i be the vertex with the deepest incoming BFS among the vertices of $BFS^{\text{out}}(w, h')$, where $h' = \min\{\tilde{h} + 1, d(w, H)\}$. The algorithm returns as an estimate $\max(d^{\text{out}}(y^o), d^{\text{out}}(w), d^{\text{in}}(y^i))$.

If $d(a, H) \leq h$, then $d^{\text{out}}(y^o)$ is at least $2h + z$ and the estimate is of the desired quality. So assume that $d(a, H) > h$, and hence $d(w, H) \geq d(a, H) \geq h + 1$. Thus $h' \geq h + 1$, as we also have $\tilde{h} \geq h$ by assumption. Assume also that $BFS^{\text{out}}(w)$ is of depth at most $2h + z - 1$ as if it is of depth at least $2h + z$ then the estimate is of the desired quality. Then, there is a vertex $w' \in BFS^{\text{out}}(w, h')$ on the shortest path from w to b with $d(w, w') = h + 1$ and hence $d(w', b) \leq h + z - 2$. As $d(a, b) = 3h + z$, we must also have $d(a, w') \geq 2h + 2$ and as $d^{\text{in}}(y^i) \geq d(a, w')$, the estimate is of the desired quality. □

Next, we analyze the running time of the algorithm.

Lemma 11. *Let $G = (V, E)$ be a graph of diameter $D = 3h + z$, where $h \geq 0$ and $z \in \{0, 1, 2\}$. If $\tilde{h} \geq h$, Approx-Diam-Sparse($G, \tilde{h}$) runs in $O(m^2/\Delta + \Delta^{\tilde{h}+1}m)$ time.*

Proof. The algorithm computes a BFS tree for every vertex of H. $|H| = O(m/\Delta)$ since there are at most that many vertices of outdegree at least Δ. Hence the BFS computation from H takes $O(m^2/\Delta)$ time.

Computing the distances of the nodes in $V \setminus H$ to H takes only $O(m)$ time. Picking the node w at largest distance to H takes $O(n)$ time. The algorithm computes $BFS^{\text{out}}(w)$ in $O(m)$ time. It then computes $BFS^{\text{in}}(v)$ for every $v \in$

$BFS^{\text{out}}(w, h')$ where $h' \leq \tilde{h} + 1$. Since we also have that $h' \leq d(w, H)$, every $v \in BFS^{\text{out}}(w, h' - 1)$ has outdegree at most Δ. Thus, $|BFS^{\text{out}}(w, h')| \leq \Delta^{h'} \leq \Delta^{\tilde{h}+1}$. The running time of computing $BFS^{\text{in}}(v)$ for every $v \in BFS^{\text{out}}(w, h')$ is hence at most $O(m\Delta^{\tilde{h}+1})$. □

We now prove Theorem 7 from the introduction.

Reminder of Theorem 7 *There is an $\tilde{O}(m^{2-1/(2h+3)})$ time deterministic algorithm that computes an estimate $\hat{D}$ with $\lceil 2D/3 \rceil \leq \hat{D} \leq D$ for all m-edge unweighted graphs of diameter $D = 3h + z$ with $h \geq 0$ and $z \in \{0, 1, 2\}$. In particular, $\hat{D} \geq 2h + z$.*

Proof. In $O(m)$ time we can get a 2-approximation to the diameter, i.e. an estimate E with $D/2 \leq E \leq D$. Since $D = 3h + z$, we have that $(E - 2)/3 \leq h \leq 2E/3$. Setting $\tilde{h} = 2E/3$ guarantees that $h \leq \tilde{h} \leq 2h + 4/3 < 2h + 2$, and hence $h \leq \tilde{h} \leq 2h + 1$.

The quality of the estimate follows from Lemma 10 and by Lemma 11, the runtime is $O(m^2/\Delta + m\Delta^{2h+2})$. Picking $\Delta = m^{1/(2h+3)}$ minimizes the running time at $O(m^{2-1/(2h+3)})$. □

Acknowledgements.

The first author wants to thank Edith Cohen, Haim Kaplan and Yahav Nussbaum for fruitful discussions on the problem. The second author wants to thank Bob Tarjan for asking whether there is an almost linear time approximation scheme for the diameter, and Ryan Williams for many discussions on the hardness of diameter and SETH.

6. REFERENCES

[1] 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.

[2] R. Albert, H. Jeong, and A.L. Barabasi. Diameter of the world wide web. *Nature*, 401:130 – 131, 1999.

[3] S. Baswana, V. Goyal, and S. Sen. All-pairs nearly 2-approximate shortest paths in $o(n^2 poly \log n)$ time. *Theor. Comput. Sci.*, 410(1):84–93, 2009.

[4] S. Baswana and T. Kavitha. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. *SIAM J. Comput.*, 39(7):2865–2896, 2010.

[5] 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.

[6] P. Berman and S. P. Kasiviswanathan. Faster approximation of distances in graphs. In *Proc. WADS*, pages 541–552, 2007.

[7] A. Bernstein. A nearly optimal algorithm for approximating replacement paths and k shortest simple paths in general graphs. In *Proc. SODA*, pages 742–755, 2010.

[8] G. Blelloch, V. Vassilevska, and R. Williams. A new combinatorial approach to sparse graph problems. In *Proc. ICALP*, pages 108–120, 2008.

[9] K. Boitmanis, K. Freivalds, P. Ledins, and R. Opmanis. Fast and simple approximation of the diameter and radius of a graph. In *WEA*, pages 98–108, 2006.

[10] T. M. Chan. All-pairs shortest paths for unweighted undirected graphs in $o(mn)$ time. *ACM Transactions on Algorithms*, 8(4):34, 2012.

[11] V. Chepoi, F. Dragan, and Y. Vaxès. Center and diameter problems in plane triangulations and quadrangulations. In *Proc. SODA*, pages 346–355, 2002.

[12] 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.

[13] E. Cohen and U. Zwick. All-pairs small-stretch paths. *J. Algorithms*, 38(2):335–353, 2001.

[14] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Computation*, 9(3):251–280, 1990.

[15] D.G. Corneil, F.F. Dragan, M. Habib, and C. Paul. Diameter determination on restricted graph families. *Discr. Appl. Math.*, 113:143 – 166, 2001.

[16] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlstrom. On problems as hard as CNFSAT. In *Proc. CCC*, pages 74–84, 2012.

[17] M. Cygan, H. N. Gabow, and P. Sankowski. Algorithmic applications of baur-strassen's theorem: Shortest cycles, diameter and matchings. In *Proc. FOCS*, 2012.

[18] D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29(5):1740–1759, 2000.

[19] D. Dvir and G. Handler. The absolute center of a network. *Networks*, 43:109 – 118, 2004.

[20] M. Elkin. Computing almost shortest paths. *ACM Transactions on Algorithms*, 1(2):283–323, 2005.

[21] S.L. Hakimi. Optimum location of switching centers and absolute centers and medians of a graph. *Oper. Res.*, 12:450 – 459, 1964.

[22] E. A. Hirsch. Two new upper bounds for SAT. In *Proc. SODA*, pages 521–530, 1998.

[23] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62:367–375, 2001.

[24] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63:512–530, 2001.

[25] F. Le Gall. Faster algorithms for rectangular matrix multiplication. In *Proc. FOCS*, 2012.

[26] B. Monien and E. Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, 10(3):287 – 295, 1985.

[27] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364, 2005.

[28] R. Paturi, P. Pudlák, and F. Zane. Satisfiability coding lemma. *Chicago J. Theor. Comput. Sci.*, 1999, 1999.

[29] M. Pătraşcu and R. Williams. On the possibility of faster SAT algorithms. In *Proc. SODA*, pages 1065–1075, 2010.

[30] I. Schiermeyer. Solving 3-satisfiability in less then 1.579^n steps. In *CSL*, pages 379–394, 1992.

[31] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proc. FOCS*, pages 410–414, 1999.

[32] R. Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *J. Comput. Syst. Sci.*, 51(3):400–403, 1995.

[33] A. Stothers. On the complexity of matrix multiplication. *Ph.D. Thesis, U. Edinburgh*, 2010.

[34] V. Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proc. STOC*, 2012. To appear.

[35] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[36] U. Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM*, 49(3):289–317, 2002.