Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs [*]

Amir Abboud
Stanford University
abboud@cs.stanford.edu

Virginia Vassilevska Williams
Stanford University
virgi@cs.stanford.edu

Joshua Wang
Stanford University
joshua.wang@cs.stanford.edu

Abstract

The radius and diameter are fundamental graph parameters, with several natural definitions for directed graphs. Each definition is well-motivated in a variety of applications. All versions of diameter and radius can be solved via solving all-pairs shortest paths (APSP), followed by a fast post-processing step. However, solving APSP on n-node graphs requires $\Omega(n^2)$ time even in sparse graphs.

We study the question: when can diameter and radius in *sparse* graphs be solved in truly subquadratic time, and when is such an algorithm unlikely? Motivated by our conditional lower bounds on computing these measures exactly in truly subquadratic time, we search for *approximation* and *fixed parameter subquadratic* algorithms, and alternatively, for reasons why they do not exist.

We find that:

- Most versions of Diameter and Radius can be solved in truly subquadratic time with *optimal* approximation guarantees, under plausible assumptions. For example, there is a 2-approximation algorithm for directed Radius with one-way distances that runs in $\tilde{O}(m\sqrt{n})$ time, while a $(2-\delta)$-approximation algorithm in $O(n^{2-\varepsilon})$ time is considered unlikely.
- On graphs with treewidth k, we can solve all versions in $2^{O(k\log k)}n^{1+o(1)}$ time. We show that these algorithms are near *optimal* since even a $(3/2-\delta)$-approximation algorithm that runs in time $2^{o(k)}n^{2-\varepsilon}$ would refute plausible assumptions.

Two conceptual contributions of this work that we hope will incite future work are: the introduction of a *Fixed Parameter Tractability in P* framework, and the statement of a differently-quantified variant of the Orthogonal Vectors Conjecture, which we call the *Hitting Set Conjecture*.

[*]The full version of the paper can be found at: http://arxiv.org/abs/1506.01799. A.A and V.V.W were supported by NSF Grants CCF-1417238 and CCF-1514339, and BSF Grant BSF:2012338. J.W was supported by Stanford Graduate Fellowship.

1 Introduction

1.1 Overview. A successful and exciting line of research in recent years attempts to understand the computational complexity of fundamental problems by relating them to each others via reductions. Certain plausible conjectured lower bounds for fundamental key problems are shown to imply that the current algorithms for many other important problems are essentially optimal. One example of a popular conjecture is the Strong Exponential Time Hypothesis (SETH) asserting that for CNF-SAT on constant width clauses, there is essentially no faster algorithm than the one that brute-forces over all possible variable assignments. Exciting consequences of SETH include a tight $n^{2-o(1)}$ lower bound for computing the diameter of a sparse n-node graph [47], the Edit Distance of two n-length sequences [8], and a tight $m^{1-o(1)}$ bound per update for maintaining single source reachability in dynamic graphs m-edge graphs [4]. This research is motivated both by the desire for a better understanding of (polynomial time) computation and by the search for practical algorithms. While classically, any polynomial time problem was considered tractable, this is no longer the case with today's enormous inputs where even quadratic time might be considered intractable.

In this work, we study three general directions for this line of work. We demonstrate their effectiveness on the fundamental problems of computing the radius and diameter of a graph, and more generally, all eccentricities.

- The first framework we introduce is *Fixed Parameter Tractability in P*. One of the most active areas of research in theoretical computer science in the past decade is *parameterized* or *multivariate* complexity [34, 37, 44]. The central idea is to study the complexity of an NP-hard problem not only in terms of the input size n but also in terms of an additional natural parameter k. This led to the development of *fixed parameter* algorithms, with running times of the form $f(k)\cdot n^{O(1)}$, for many fundamental problems. Such problems are then called fixed parameter tractable (FPT).

  When considering problems that already have polynomial time algorithms, the usual FPT definition is not very interesting since all problems in P are fixed parameter tractable with respect to any parameter. We propose a more fine-grained approach: Since quadratic time is a bottleneck in many applications, we propose to treat it as *intractable*, just as super-polynomial time is traditionally treated. We seek interesting *Fixed Parameter Subquadratic* algorithms, with running time of the form $O(f(k) \cdot n^{2-\varepsilon})$ for some $\varepsilon > 0$. For example, we will consider natural parameterizations of Radius and Diameter such as the treewidth tw of the input graph, and ask whether there is an $O(f(tw) \cdot n^{2-\varepsilon})$ time algorithm for the problems, and if so, for what functions f. The more general approach is, given a problem that is in $O(n^c)$ time for some c but is believed to also require $n^{c-o(1)}$ time, study for which parameters k and functions f, the problem has an $O(f(k)n^{c-\varepsilon})$ time algorithm for some $\varepsilon > 0$, and if possible, also prove matching (conditional) lower bounds on f.

- The second direction we follow is that of *faster approximability in P*, which had already been studied in previous works [47, 25, 20]. We consider problems that can be solved exactly in polynomial time, say $O(n^2)$ time, for which a conditional $n^{2-o(1)}$ lower bound is known, and we ask: can we solve the problem approximately in truly subqudratic time? If so, what is the best approximation ratio we can obtain in $O(n^{2-\varepsilon})$ time for constant $\varepsilon > 0$ ("truly subquadratic") time? More generally, these questions can be asked for problems regardless of whether they are in P or not: suppose that a problem A can be solved exactly in $a(n)$ time; for what values of α, is there an α-approximation algorithm that runs in $a(n)^{1-\varepsilon}$ time for $\varepsilon > 0$?

- The third approach we propose is to look at *different quantifications* of conjectured-to-be-hard problems. For example, a problem that requires $n^{2-o(1)}$ under SETH is the orthogonal vectors (OV) problem: given two lists A and B, each containing n d-dimensional vectors each (for $d = \omega(\log n)$), does there exist a pair of vectors $a \in A, b \in B$ such that $a \cdot b = 0$? This OV problem is the basis for almost all hardness reductions for problems in P. The quantifiers in the OV problem are $\exists\,\exists$ (i.e. $\exists a \exists b$). A different quantification of the problem is $\forall\,\exists$, giving the Hitting Set Existence problem: is it true that *for all* vectors a in A, there *exists* a vector in B that is orthogonal to a?

These new quantified versions could be: (1) intuitively as hard as the original problem, and (2) much more appropriate for proving hardness for other problems for which it seems very hard or even provably unlikely to prove hardness by reduction from the original problem (because of type mismatch).

This approach allows us to prove tight conditional lower bounds for approximating the Radius of a sparse graph, a task that has been elusive despite many efforts, even though such a lower bound for the very similar Diameter problem was proven a few years back [47]. To achieve this, we introduce a new natural and plausible variant of the OV conjecture that OV requires $n^{2-o(1)}$ time. This new Hitting Set Conjecture asserts that the Hitting Set problem also requires $n^{2-o(1)}$ time.

1.2 Case study: Diameter and Radius. We study two of the most basic graph parameters: radius and diameter. The diameter of an undirected graph is the longest distance, and the radius is the shortest distance from a node to the furthest node from it. Intuitively, the node that achieves the radius, the so-called *center* of the graph, is "close" to all other nodes. In directed graphs, depending on the application, there may be multiple definitions for "closeness": a node can be close in the sense that it has short paths *to* other nodes ("source"), *from* other nodes ("target"), or even to and then back from other nodes ("roundtrip"). That is, there are several natural definitions of both radius and diameter for directed graphs; all of them are well-studied [28, 40, 27, 36, 6, 30, 26, 35, 11, 12, 57, 58, 23, 38, 54, 47, 25, 2, 17] (and many others). Even estimating the diameter and radius of a network efficiently is useful in practical applications (e.g. the analysis of social networks) and serves as a basic primitive.

Although the problems are very well-studied, theoretically the fastest known exact algorithms for both Diameter and Radius compute all pairs shortest paths (APSP) and then run a fast postprocessing procedure. Unfortunately, any algorithm for APSP necessarily takes $\Omega(n^2)$ time in n-node graphs *regardless of the sparsity*, since the output is quadratic. However the output for both Radius and Diameter is a single integer, and it is far from obvious why $\Omega(n^2)$ time in sparse graphs (say with, $O(n)$ edges) would be necessary. In this paper we address the question below, providing both algorithms and lower bounds.

When can Diameter and Radius in sparse graphs be solved in $O(n^{2-\varepsilon})$ time for $\varepsilon > 0$?

The study of the above question has a clear practical motivation: quadratic time on real-world graphs is infeasible, so that the boundary between the tractable and intractable is really in the low-polynomial regime.

In the rest of this paper, we say that a bound is *subquadratic* if it can be bounded by $O(n^{2-\varepsilon})$ for some $\varepsilon > 0$, while upper bounds of the form $n^{2-o(1)}$ are only *mildly subquadratic*.

Barriers. Recent work has revealed convincing evidence that solving Diameter in subquadratic time might not be possible, even in undirected graphs. Roditty and Vassilevska W. [47] showed that an algorithm that can distinguish between diameter 2 and 3 in an undirected sparse graph in subquadratic time refutes the following widely believed conjecture.

The Orthogonal Vectors Conjecture: There is no $\varepsilon > 0$ such that for all $c \geq 1$, there is an algorithm that given two lists of n boolean vectors $A, B \subseteq \{0,1\}^d$ where $d = c \log n$ can determine if there is an orthogonal pair $a \in A, b \in B$, in $O(n^{2-\varepsilon})$ time.

The problem in the above conjecture is called the Orthogonal Vectors (OV) problem. The best known algorithm for it runs in mildly subquadratic $n^{2-1/O(\log (d/\log n))}$ time [3]. Williams [55] showed that the OV conjecture is implied by the well-known Strong Exponential Time Hypothesis (SETH) of Impagliazzo, Paturi and Zane [42, 41]. Nowadays many papers base the hardness of problems on SETH and the OV conjecture. This holds both for NP-hard problems (e.g. [33]), as well as problems in P [46, 4, 5, 18, 8, 1, 19].

For the Radius problem, the only known barriers to solving the problem exactly are based on other conjectures. Recent work [2] shows that if the radius of a possibly dense graph can be computed in truly subcubic time, $O(n^{3-\varepsilon})$ for $\varepsilon > 0$, then APSP also admits a truly subcubic algorithm. Such an algorithm for APSP has long eluded researchers, and it is often conjectured that it does not exist (e.g. [56, 4, 49, 51]). For dense graphs the latter result essentially settles the question of computing Radius exactly. For sparse graphs, however, only a much weaker result is known: any $T(m)$ time algorithm for the radius of an m-edge graph can be used to find a triangle in an m-edge graph in $O(T(m))$ time [2]. The limit of current techniques for triangle finding is $O(m^{4/3})$ [7] (if the matrix multiplication exponent is 2), and hence this result gives some reason to believe that obtaining a very fast algorithm for Radius in sparse graphs would be hard. Nevertheless, this result says nothing about the existence of an $O(n^{2-\varepsilon})$ time algorithm.

A natural approach to prove Radius limitations in sparse graphs is to base them on the OV conjecture. However, such a lower bound has remained elusive [2, 16]. This is due to the following type mismatch. The OV problem asks for the existence of a pair of vectors with a certain property, just as Diameter asks for the existence of a pair of nodes that are far, i.e. both are of type $\exists x \exists y$. Meanwhile, Radius asks for the existence of a node such that *all* nodes are close, i.e. $\exists x \forall y$. This quantifier disagreement is the difficulty of proving a lower bound based on OV, and suggests the following natural and plausible variant of the OV conjecture.

The Hitting Set Conjecture: There is no $\varepsilon > 0$ such that for all $c \geq 1$, there is an algorithm that given two lists A, B of n subsets of a universe U of size $c \log n$, can decide in $O(n^{2-\varepsilon})$ time if there is a set in the first list that intersects every set in the second list, i.e. a "hitting set".

We call the problem in this conjecture the Hitting Set Existence (HSE) problem. An equivalent version of the HSE problem is as follows: given two lists $A, B \subseteq \{0,1\}^d$, determine whether there is a vector $a \in A$ that is not orthogonal to any vector $b \in B$. The HSE problem can also be solved in mildly subquadratic $n^{2-1/O(\log (d/\log n))}$ time [3], where $d = |U|$. The HS conjecture is an offline version of folklore conjectured lower bounds on the hardness of classic online problems such as set intersection and partial match studied for instance by Patrascu [45].

We discuss these conjectures in full version and also show that the OV conjecture is implied by the HS conjecture.

With the following theorem, we complete the picture (at least conditionally) for the exact computation of Radius and Diameter in undirected sparse graphs.

THEOREM 1.1. *If for some $\varepsilon > 0$, there is an algorithm that can determine if a given undirected, unweighted graph with n nodes and $O(n)$ edges has radius 2 or 3 in $O(n^{2-\varepsilon})$ time, then the HS Conjecture is false.*

Subsequent work of Carmoniso et al [22] gave evidence that basing the HS conjecture on SETH is unlikely. Using their framework one can show that SETH-hardness for Radius is similarly unlikely. This can be viewed as justification for the introduction of a new conjecture into the web of reductions in P.

Overcoming the barriers. The rest of the paper tries to obtain meaningful positive results that overcome the barriers above. We consider two of the most successful approaches for coping with NP-hard problems: *approximation* and *parameterization*. In the first approach, we will address questions of the form: what is the smallest constant c such that we can get a c-approximation algorithm for Diameter and Radius in directed and undirected graphs in $O(n^{2-\varepsilon})$ time? In the second approach, we will consider natural parameterizations of Radius and Diameter such as the treewidth of the input graph, and ask whether there is an $O(f(tw) \cdot n^{2-\varepsilon})$ time, or *fixed parameter subquadratic*, algorithm for the problems, and if so, for what functions f.

The positive results we obtain in the two parts of our work (corresponding to the two approaches) will use a disjoint set of tools. However, in both approaches, the upper bounds will be matched (or nearly matched) by lower bounds that are obtained from similar constructions.

1.3 Approximation algorithms

In undirected graphs, both Diameter and Radius can be 2-approximated by a simple linear time algorithm: pick any node and report the largest distance from it. Aingworth et al. [6] obtained an $\tilde{O}(n^2 + m\sqrt{n})$ time almost-3/2-approximation algorithm for Diameter and Radius in undirected graphs. Roditty and Vassilevska W. [47] obtained a randomized almost-3/2-approximation algorithm with runtime $\tilde{O}(m\sqrt{n})$, and

Chechik et al. [25] derandomized the algorithm and obtained a genuine $3/2$-approximation algorithm running in time $\tilde{O}(mn^{2/3})$. As previously mentioned, [47] also showed that any $O(n^{2-\varepsilon})$ time algorithm that $(3/2-\delta)$-approximates the diameter (for $\varepsilon, \delta > 0$) breaks the OV conjecture (as it would distinguish between graphs of diameter 2 and 3). We show that the known approximation algorithms for Radius are also likely tight. An immediate corollary of Theorem 1.1 is:

COROLLARY 1.1. *A subquadratic $(3/2-\delta)$-approximation algorithm for* UNDIRECTEDRADIUS*, for some $\delta > 0$, refutes the Hitting Set conjecture.*

The eccentricity of a node is the largest distance out of it. Diameter is the maximum eccentricity, and radius is the minimum. Even though both undirected Diameter and Radius can be $3/2$-approximated in subquadratic time, the best known subquadratic algorithm for estimating *all* the eccentricities, by Chechik et al. [25], only gives a $5/3$ approximation. We show that this result is tight conditioned on the OV conjecture[1].

THEOREM 1.2. *A $(5/3-\delta)$ approximation algorithm for the eccentricities of all nodes in undirected sparse graphs that runs in subquadratic time refutes the Orthogonal Vectors Conjecture.*

This completes the picture for undirected graphs and we now turn our attention to directed graphs, where much less was known before our work. To better highlight the novelty of this work, we will only present our results for Radius on directed graphs (see Table 1). Our results for Diameter can be found in Table 2.

One-way distances. The first definition of Radius on directed graphs, Source Radius, is the natural extension of the undirected Radius definition: $\min_x \max_v d(x,v)$. While in undirected graphs a 2-approximation is trivial, this is no longer the case for directed graphs. In undirected graphs, we can claim for any center v^*, arbitrary u, and for all x, by the triangle inequality, $d(u,x) \leq d(u,v^*) + d(v^*,x) = d(v^*,u) + d(v^*,x) \leq 2R$. Since in directed graphs $d(u,v^*)$ is unrelated to $d(v^*,u)$, no approximation is guaranteed. Even the known $3/2$-approximation algorithms [47, 25] do not work since for directed graphs all that they can guarantee is that they compute the eccentricity of some node u with either $d(u,v^*) \leq R/2$ or $d(v^*,u) \leq R/2$, and in the latter case no approximation can be guaranteed. Our first algorithmic contribution is a new subquadratic 2-approximation algorithm for Source Radius overcoming the above issues with a two level sampling approach.

[1]Independently, Cairo, Grossi, and Rizzi proved a similar lower bound under SETH (private communication).

THEOREM 1.3. *Given a directed unweighted graph on n nodes and m edges, there is an algorithm that outputs R^* such that $R \leq R^* \leq 2R$, and runs in time $O(m\sqrt{n}\log^2 n)$.*

Our algorithm is lightweight and easy to implement. Theorem 1.1 implies that a subquadratic algorithm for Source Radius is not likely to have an approximation guarantee better than $3/2$ and makes one wonder whether a $3/2$ guarantee is possible in subquadratic time, as is the case in undirected graphs. However, using the directed edges we manage to increase the gap in the lower bound construction and prove that the approximation factor of our algorithm is optimal for a subquadratic algorithm under the HS conjecture.

THEOREM 1.4. *A $(2-\delta)$-approximation algorithm for Source Radius in sparse graphs that runs in subquadratic time refutes the Hitting Set Conjecture.*

Roundtrip and longest distances. The roundtrip distance between u and v is the distance from u to v plus the distance from v to u, i.e. the sum of both one-way distances. The Roundtrip Radius of the graph is $\min_x \max_v d(x,v) + d(v,x)$. The Max-distance between u and v is the largest of the two one-way distances. The Max Radius of the graph is $\min_x \max_v \max\{d(x,v), d(v,x)\}$.

These definitions are natural ways to turn the distances in directed graphs into a metric. This means that by picking any node as the center we obtain a 2-approximation near-linear time algorithm for Roundtrip Radius and Max Radius. Moreover, Cowen and Wagner [31, 32] observed that many of the techniques for approximating distances in undirected graphs can be adapted to handle roundtrip distances, which also led to the roundtrip-spanners of Roditty, Thorup, and Zwick [48]. This seems to suggest that these versions of Radius should be more like the undirected version where a $3/2$-approximation is possible in subquadratic time, and not like Source Radius where the 2 factor is tight. Quite surprisingly, via a delicate reduction, we were able to obtain a gap of 2 in the lower bound constructions, and show that anything better than the trivial 2-approximation is unlikely to run in subquadratic time.

THEOREM 1.5. *A $(2-\delta)$-approximation algorithm for Roundtrip Radius or Max Radius that runs in $O(m^{2-\varepsilon})$ time on sparse graphs, for some $\varepsilon, \delta > 0$, refutes the Hitting Set Conjecture.*

Min Radius. Finally, we consider a less standard but quite intriguing variant of Radius where distance is the shorter of the two directions. Formally, we define the Min-eccentricity of a node u to be the maximum over nodes v of $\min\{d(u,v), d(v,u)\}$. The node with minimum Min-eccentricity is the Min-Center of the graph and its

Min-eccentricity is the Min-radius. This directed definition naturally models certain applications. For example, in a network representing geographic locations, the Min-center would be the optimal location to place a hospital since it will allow for the fastest possible medical treatment (either by driving to the hospital or by having an ambulance drive from the hospital to the patient) for any location in the graph. This is the only directed Radius version without a trivial linear time algorithm on a DAG[2].

Although the problem becomes easy once we compute APSP, it is quite challenging to approximate to within any constant factor without knowing all the distances. Intuitively, a node with Min-eccentricity R could be very hard to distinguish from nodes that have infinite min-distance to a single node in the graph. We give a linear time algorithm for this simpler task.

PROPOSITION 1.1. *There is an $O(m)$ time algorithm that can check if there is a node in a directed graph with m edges that can reach or be reached from any other node. Consequently, there is a factor n approximation for Min-Radius in linear time.*

Finally, we consider approximation algorithms for Min-Radius on a DAG - which, in our opinion, is the most natural version of the question "what is the center of a DAG"? We devise a recursive 3-approximation subquadratic algorithm for the problem and show that a better than 2 factor is unlikely.

THEOREM 1.6. *There is a 3-approximation algorithm for Min-Radius on n node, m edge DAGs that runs in $O(m\sqrt{n}\log n)$ time, and a subquadratic $(2-\delta)$ approximation algorithm that runs in subquadratic time on sparse DAGs refutes the Hitting Set Conjecture.*

1.4 Fixed Parameter Subquadratic Algorithms

Treewidth. We will illustrate our approach using the Diameter problem on n-node undirected graphs of treewidth k. This is one of the most popular parameterizations of graph problems in the literature on parameterized complexity, and is usually considered when the problem becomes easy on trees [13]. Note that a folklore algorithm solves Diameter in $\tilde{O}(n)$ on trees: do Dijkstra's from an arbitrary node u, and then Dijkstra's from the furthest node v from u, report the largest distance found. Since in arbitrary graphs (where the treewidth is $\leq n$) one can solve Diameter in $\tilde{O}(n^2)$ time, a natural conjecture is that the right runtime bound in terms of treewidth is $O(kn)$. Unfortunately, we observe that the lower bound construction for Diameter rules out any such algorithm. In fact, it shows that in any fixed parameter subquadratic running time for Diameter, the dependence on k, the treewidth, must be exponential!

THEOREM 1.7. *If for some $\varepsilon > 0$, there is an algorithm that can distinguish between diameter 2 and 3 in an undirected unweighted graph of treewidth k in $2^{o(k)} \cdot n^{2-\varepsilon}$ time, then the Orthogonal Vectors Conjecture is false. If such an algorithm exists for Radius, then the Hitting Set Conjecture would be false.*

This lower bound is quite surprising when contrasted with the near-linear time $(1+\varepsilon)$-approximation algorithm for Diameter in planar graphs of Weimann and Yuster [54], since it shows that such a result is unlikely on non-planar graphs of treewidth $\Theta(\log n)$. Furthermore, our lower bound also applies to graphs of *pathwidth* k. On the positive side, this bound led us to look for a $2^{O(k)}n^{2-\varepsilon}$ time algorithm for Diameter.

Although computing the treewidth is an NP-hard problem, Bodlaender et al. [15] obtained a $2^{O(k)}n$ time algorithm (fixed parameter linear time) that returns a tree decomposition of bag size $O(k)$. This gives hope that the known techniques from FPT algorithms will give interesting subquadratic algorithms. For example, we could apply Courcelle's theorem to solve Diameter in $f(k) \cdot n$ time, for some huge but computable $f(k)$. Instead, we use a technique that, to our knowledge, was never used for obtaining FPT algorithms for NP-hard problems and obtain a fixed parameter subquadratic algorithm for Diameter and Radius parameterized by treewidth that almost matches our lower bound.

THEOREM 1.8. *There is an algorithm that solves Diameter and Radius* exactly *in undirected graphs of treewidth k in $2^{O(k\log k)} \cdot n^{1+o(1)}$ time.*

Closing the gap in the dependence on k between the $2^{O(k\log k)} \cdot n^{1+o(1)}$ upper bound and the $2^{o(k)} \cdot n^{2-\varepsilon}$ conditional lower bound is a very interesting open question. In Section 2 we also obtain exact algorithms with similar upper bounds for all versions of directed Radius and Diameter that we consider in this work. We can also compute all the eccentricities of the graph in the same time.

Besides utilizing the tree decomposition to find separators, the main tool in our algorithms is a reduction to an orthogonal range query problem and then using known data structures to answer queries efficiently. This technique was used by Cabello and Knauer [21] to obtain near-linear time algorithms for computing the Wiener index of a fixed treewidth graph[3].

The exact running time of our algorithm for Diameter is $O(k^2 n \log^{k-1} n)$ and we believe it can be a practical

[2]The Max and Roundtrip Radius are infinite on a DAG, and the Source Radius is the eccentricity of the first node in the topological order.

[3]The Wiener index of a graph is the sum of distances. It can be computed in $O(mn)$ time in general graphs using APSP.

alternative to known Diameter algorithms when a good bound on the treewidth of the graph is known. It is known that many real-life networks are tree-like (see [14] and the surveys therein.)

Other parameters. Perhaps more basic parameterizations for Diameter would be: D - the diameter of the graph, and Δ - the maximum degree of a node in the graph. Unfortunately, the lower bound constructions show that these cases are not fixed parameter subquadratic. It hard to solve Diameter in subquadratic time even when the diameter is 3, and there is a simple reduction from Diameter on a sparse graph on n nodes of arbitrary max degree to Diameter on a constant degree graph on $O(n)$ nodes. The same holds for Radius under the HS conjecture. Thus, Radius and Diameter are not fixed parameter subquadratic when parameterized by the degree or the diameter of the input graph, unless our conjectures fail.

Related work. Most related to our parameterized complexity results are known algorithms for Diameter and Radius on special classes of graphs, e.g. [40, 27, 36, 30, 26, 35, 11, 12, 57, 54]. Our two dimensional complexity results, however, show how the complexity changes as the input graph becomes "more complicated". Independently of our work, Giannopoulou, Mertzios, and Niedermeier [39] also propose the study of parameterized complexity for problems in P. The authors suggest searching for $f(k) \cdot n^c$ algorithms for problems for which the best known algorithm takes $O(n^d)$ time, where d is much larger than c. Their case study is the problem of finding the longest path in an interval graph, and they improve the known $O(n^4)$ time algorithm to $f(k) \cdot n$ for a certain natural parameter k. We are not aware of previous negative parameterized complexity results for problems in P.

1.5 Extensions

In the full version of the paper we show that there is a *subquadratic equivalence* between the OV problem and the problem of distinguishing between diameter 2 and 3 in sparse graphs, in the sense that a subquadratic algorithm for one implies a subquadratic algorithm for the other. Similarly, there is a subquadratic equivalnce between the HS problem and distinguishing between radius 2 and 3 in sparse graphs. To prove the equivalence we devise new reductions *from* the graph problems *to* OV and HS, via a low-degree high-degree analysis and a hashing trick. From the mildly subquadratic algorithms for OV and HS [3], we obtain new mildly subquadratic algorithms for radius and diameter.

THEOREM 1.9. *There is an algorithm that can decide whether the diameter (or radius) of a given sparse graph is* 2 *or* 3*, in* $n^2/2^{\Omega(\sqrt{\log n})}$ *time.*

This result shows that on sparse 3-layered graphs, there is a superpolylogarithmic gap between the complexities of diameter and APSP, since there is an unconditional $\Omega(n^2)$ lower bound for APSP. Such gaps were only known for special classes of graphs (like bounded treewidth graphs), while it is known that the 3-layered case is typically the hardest for APSP.

Finally, we demonstrate the potential of the HS conjecture for explaining the hardness of other problems by proving a new conditional lower bound for computing the *median* of the graph. In undirected graphs, the median is the node v that minimizes the sum of distance to all the other nodes $\sum_u d(v,u)$. Finding the median is equivalent to finding the node with largest *closeness centrality* in the graph [9, 10, 50] - a very important task in network analysis [40, 52]. Like Radius, it was known that computing the median of dense weighted graphs in subcubic time refutes the APSP conjecture [2] while no consequences of a subquadratic algorithm in sparse graphs were known. In stark contrast to Radius, however, Median is known to have a near-linear time $(1+\varepsilon)$ approximation [43, 53, 29]. It turns out that the HS conjecture implies that this subquadratic running is impossible if we want to know the median exactly.

THEOREM 1.10. *A subquadratic algorithm for finding the median of a sparse unweighted undirected graph refutes the Hitting Set Conjecture.*

2 Subquadratic Approximation Algorithms

In this section, we cover our approximation algorithms for SOURCERADIUS, MINDIAMETER, and MINRADIUS.

Source Radius Although it was trivial to find a 2-approximation in the UNDIRECTEDRADIUS, ROUNDTRIPRADIUS, and MAXRADIUS problems, the nonsymmetric nature of SOURCERADIUS makes it nontrivial to achieve a 2-approximation. Choosing an arbitrary vertex as before may yield an infinitely bad approximation factor, if it cannot reach the rest of the graph.

Arbitrary vertices worked before since we could reach a center v^* within R and then any other node within another R. Hence a natural attempt is to try to find a vertex that can reach the center within R. Let $Pre(v,\ell)$ be the set of nodes that can reach v within ℓ. If $Pre(v^*,R)$ was large, we could use a standard hitting set argument to find a member. This observation reduces the problem to one where $Pre(v^*,R)$ is small.

We next make the observation that the center must show up in every $Pre(v,R)$. Hence, if we could find any small $Pre(v,R)$, we could run forward Dijkstra's from its nodes. One way of figuring out which $Pre(v,R)$ are small is to use the fact that searching for the closest k nodes from a starting node can be done with a modified Dijkstra in $O(k^2 \log n)$ time, given that edge costs at each node are already sorted (for each of at most k searched nodes, we only need to enqueue the cheapest edge which does not go to a searched node, which can be done in k^2 updates).

However, these short Dijkstra's from every node will incur a $\tilde{O}(n^2)$ cost (all of our thresholds are roughly $\sqrt{n}$). Instead, we can be more clever with how we use our hitting set. With high probability, the hitting set hits every large $Pre(v, R)$. Hence any $Pre(v, R)$ not hit must be small. At least one of them must not be hit, since we assumed $Pre(v^*, R)$ was not.

If we knew the radius, these ideas would give us a running time of $O(m\sqrt{n}\log n)$. However, doing a binary search for the radius incurs an additional $(\log Mn)$ factor. To avoid this, we use an idea from the Aingworth et al. seminal algorithm for a $\frac{2}{3}$-approximation of undirected diameter; choosing the furthest node from the hitting set simulates locating a small $Pre(v^*, R)$ for every R simultaneously.

THEOREM 2.1. *There is a $O(m\sqrt{n}\log^2 n)$-time Monte Carlo algorithm that approximates* SOURCERADIUS *on a graph G within a factor of* 2.

Proof. We claim that algorithm 1 has the desired properties:

Algorithm 1: ApproximateSourceRadius(G, R)

```
Sample a hitting set $S_1$ of $O(\sqrt{n}\log n)$ nodes;
Run forward Dijkstra's from all $s \in S_1$;
Let $w \in V$ maximize $\min_{s\in S_1} d(s, w)$. Run a reverse
Dijkstra from $w$;
Let $S_2$ be the $\sqrt{n}$ closest nodes to $w$. Run forward
Dijkstra's from all $s \in S_2$;
return the best source-eccentricity of all nodes in
$S_1 \cup S_2$;
```

This algorithm relies on the fact that with high probability, for any set X of the closest $\sqrt{n}$ nodes to a given node in the graph, our randomly-chosen hitting set S_1 will intersect X (note that there are only O(n) possible such X given our graph). This follows from a standard argument.

Now we can prove the claimed approximation guarantee of our algorithm. If some $s \in S_1$ can reach the center within R, we are done. Otherwise, v^* is more than R away from S_1, and hence w is as well. Since S_2 has $\sqrt{n}$ nodes, it intersects S_1 w.h.p. since S_1 is a hitting set (we want it to hit, for each node, the closest $\sqrt{n}$ nodes going backwards). Suppose v is in this intersection, then $d(v, w) > R$. But then S_2 is defined by how close nodes are to w, it must contain all nodes that can reach w in less than R. This includes v^*, so we are done.

Now we compute the running time of this algorithm. Running Dijkstras from every node in S_1 takes $O(m\sqrt{n}\log^2 n)$ time. Running Dijkstra from w takes $O(m\log n)$ time. Finally, running Dijkstras from S_2 which has $\sqrt{n}$ nodes takes $O(m\sqrt{n}\log n)$ time. This completes the proof.

MINDIAMETER We next present an algorithm for MINDIAMETER on general graphs.

LEMMA 2.1. *Given $\epsilon \geq 0$, there is a $\tilde{O}(mn^{1-\epsilon})$-time algorithm that approximates* MINDIAMETER *on directed graphs within a factor of n^ϵ.*

Proof. Suppose that the diameter is realized by the pair of points (u^*, v^*) where $d(u^*, v^*) = D$ and $d(v^*, u^*) \geq D$. If D is at most n^ϵ, then any edge is a sufficient approximation.

Consider the case where D is larger than n^ϵ. We choose a hitting set S of $\tilde{O}(n^{1-\epsilon})$ nodes that, with high probability, hits the middle third of any shortest path longer than n^ϵ. In particular it hits the middle third of the shortest path from u^* to v^* at vertex w, so that $d(u^*, w) \geq \frac{D}{3}$ and $d(w, v^*) \geq \frac{D}{3}$. Notice that one of $d(v^*, w)$ or $d(w, u^*)$ must also be at least $\frac{D}{3}$ long, otherwise $d(v^*, u^*) < D$.

Hence if we Dijkstra from all nodes in S and return $\max_{s\in S, v\in V} \min\{d(s, v), d(v, s)\}$, this yields a n^ϵ-approximation. But this takes only $\tilde{O}(mn^{1-\epsilon})$ time, which completes the proof.

We get a much better algorithm for MINDIAMETER on DAGs, since we can use the topological order of the graph to run a divide-and-conquer.

THEOREM 2.2. *There is a $O(m\log n)$-time algorithm that approximates* MINDIAMETER *on a DAG G within a factor of* 2.

Proof. Since G is a DAG, we can run a topological sort in $O(m)$ time and use this order to relabel the vertices as $\{0, 1, 2, \ldots, n-1\}$ so that edges run from lower- numbered nodes to higher-numbered nodes. Suppose that the diameter is realized by the pair of points (u^*, v^*), $u^* < v^*$. There are three possible cases:

1. $u^*, v^* < \frac{n}{2}$;
2. $\frac{n}{2} \leq u^*, v^*$;
3. $u^* < \frac{n}{2} \leq v^*$.

In case (3), consider node $\frac{n}{2}$, which we denote as w. $d(u^*, v^*) \leq d(u^*, w) + d(w, v^*)$ and so either $d(u^*, w) \geq \frac{D}{2}$ or $d(w, u^*) \geq \frac{D}{2}$. Moreover, since $u^* \leq w \leq v^*$, returning $\max\{\max_{v\leq w} d(v, w), \max_{w\leq v} d(w, v)\}$ definitely yields a 2-approximation for the diameter. Note that $d(v, w)$ and $d(w, v)$ can be computed for all v with a DP in $O(m)$ time.

Otherwise, if case (3) does not hold, run the algorithm recursively on the subgraphs of G induced by the first and last $\frac{n}{2}$ nodes in topological order. Building these induced graphs takes $O(m)$ time. There are $\log n$ levels of recursion, and each level takes $O(m)$ time: 2^i DPs on $\frac{n}{2^i}$ nodes each where the total number of edges is at most m. The total time is hence $O(m\log n)$.

MinRadius MinRadius is a difficult problem on general graphs, but it turns out that we can quickly determine which vertices have a finite min-eccentricity:

Lemma 2.2. *There is a $O(m + n)$-time algorithm that determines which vertices in a directed graph G have a finite min-eccentricity.*

Proof. In linear time, we can compute the strongly connected components of G. Notice that a vertex has a finite min-eccentricity iff its corresponding vertex in the SCC graph has a finite min-eccentricity. Hence it suffices to consider the problem on DAGs.

We next compute a topological order of the vertices, which can be done in linear time. It suffices to determine which nodes can be reached by all nodes before them in the topological order, since then we could also determine which nodes can reach all nodes *after* them in the topological order by symmetry.

In order to do this, we precompute for each node, the first node in the topological order it has an edge to. This can be done in linear time by taking a minimum over all edges coming out of a node, and noting each edge is used only once.

Fix some node v. Suppose that every node before v has an edge to a node which is before v or is v. Then every node before v can reach it, since we can keep taking edges that do not take us past v, and each edge moves us forward in the DAG. On the other hand, if some node before v only has edges to nodes after v, then not every node before v can reach it, since we have a counterexample.

We will check this property by counting, for each node v, the number of nodes before v that have an edge to a node before v or to v. However, this is easy to do with our precomputation. The count is zero for the first node, and the count for the i^{th} node is the count for the previous node, plus the number of nodes whose earliest neighbor is the current node. We can use our precomputation to generate the latter values in linear time, and then compute the counts for each node in order in linear time again.

All of our computations took $O(m+n)$ time, as desired. This completes the proof.

Like MinDiameter, MinRadius turns out to be easier on DAGs since we can run a divide-and-conquer:

Theorem 2.3. *There is a $O(m\sqrt{n}(\log Mn))$-time algorithm that approximates* MinRadius *on a DAG G within a factor of* 3.

Proof. First we show that there is an algorithm that, given the radius R, finds a vertex with eccentricity at most $3R$ or guarantees all vertices have eccentricity strictly greater than R. This algorithm will run in $O(m\sqrt{n})$ time. From this claim, we can binary search for R in the range $[0, Mn]$, yielding the desired result.

Since G is a DAG, we can run a topological sort and use this order to relabel the vertices as $\{0, 1, 2, \ldots, n-1\}$ so that edges run from lower- numbered nodes to higher-numbered nodes. Notice that since G is a DAG, if we choose $u, v \in V$ with $u < v$, $d(v, u) = \infty$ so we are only concerned with $d(u, v)$. Furthermore, suppose that $d(u, v) > 2R$. We claim that the center cannot be in the interval $[u, v]$, since then there is a path from u to v through the center with length at most $2R$.

Algorithm 2 uses this observation to return a vertex with eccentricity at most $3R$ or guarantees all vertices have eccentricity strictly more than R.

Algorithm 2: ApproximateCenter(G, R)

```
Initialize a vector $A$ with $\sqrt{n}$ evenly-spaced vertices,
i.e. $A[i] = i\frac{n-1}{\sqrt{n}-1}$;
for $i = 0, 1, \ldots, \sqrt{n} - 1$ do
    Use a DP to compute $d(v, A[i])$ and $d(A[i], v)$
    for all $v \in V$;
    if $\forall v \in V$, $\min(d(v, A[i]), d(A[i], v)) \leq 2R$
    then
        return $A[i]$;
Let $S$ be a stack of vertex intervals, intially empty;
for $i = 0, 1, \ldots, \sqrt{n} - 1$ do
    Let $\ell$ be the topologically-first vertex $v$ such that
    $d(v, A[i]) > 2R$, or $A[i]$ if no vertex satisfies this
    condition;
    Let $r$ be the topologically-last vertex $v$ such that
    $d(A[i], v) > 2R$, or $A[i]$ if no vertex satisfies this
    condition;
    Suppose the top interval of $S$ is $[a, b]$. If
    $\ell \leq b + 1$, then pop $[a, b]$ and push $[a, r]$.
    Otherwise, just push $[\ell, r]$.
for adjacent vertex intervals $[a, b]$ and $[c, d]$ in $S$ do
    for $u \in [b + 1, c - 1]$ do
        Use a DP to compute $d(v, u)$ and $d(u, v)$ for
        all $v \in [a, d]$;
        if $\forall v \in [a, d]$, $\min(d(v, u), d(u, v)) \leq R$
        then
            return $u$;
return all vertices have eccentricity strictly greater
than $R$;
```

First, we will show that Algorithm 2 is correct. If it returns some node $A[i]$, then every node was within $2R$ of that node and hence it does have eccentricity at most $3R$. Otherwise, each node $A[i]$ has some node u_i that is strictly more than $2R$ away (in the appropriate, non-infinite direction). If $u_i < A[i]$, then $[u_i, A[i]]$ cannot contain a vertex of eccentricity at most R. Similarly, if $A[i] < u_i$, then $[A[i], u_i]$ cannot contain a vertex of eccentricity at most R. Hence every interval of S cannot contain a vertex of eccentricity at most R.

The next phase of the algorithm searches the regions between adjacent intervals of S (note that since the first node is in an interval of S, as well as the last node, all remaining nodes fall between two intervals of S). Suppose that some $u \in [b+1, c-1]$ can reach all $v \in [a, d]$ in at most R distance, either forward or backwards. Then consider the node of A immediately to its left, $A[i]$. u can reach $A[i]$ (backward) in at most R distance. By construction a is either the topologically-first vertex v that cannot be reached by $A[i]$ (backwards) in $2R$ distance, or lies before that (due to a union with an even earlier region). Hence u can reach *all nodes* (backwards) to the left of a with at most $3R$ distance by going through $A[i]$. Hence u can reach all nodes before it (backwards) using only $3R$ distance. Similarly, it can reach all nodes after it (forwards) using only $3R$ distance. Hence u has eccentricity at most $3R$, and is valid to return.

Otherwise, all vertices outside of intervals of S have eccentricity strictly more than R. But then every vertex has eccentricity more than R. Hence the final return statement is also correct.

Next, we analyze the running time of Algorithm 2. The first phase of our algorithm computes $\sqrt{n}$ DPs, which take $O(m)$ time each. Computing S takes $O(n\sqrt{n})$ time.

Next, we compute distances for every node not in one of S's intervals. In order to bound the running time of this phase, we note two things. Firstly, no region between intervals can contain more than $O(\sqrt{n})$ points since our inital points are all in intervals of S and we chose them to be not too far apart. Secondly, any edge only needs to be considered for at most two regions between intervals (and only then if it lies in some interval of S). Since the running time of our DPs is linear in the number of edges the DP must consider, our running total running time is bounded by $O(m\sqrt{n})$.

The total running time is hence $O(m\sqrt{n})$, as claimed.

This completes the proof.

3 Fixed Parameter Subquadratic Algorithms

In this section, we present algorithms for diameter and radius on graphs of small treewidth. We begin with the undirected case, which illustrates the technique, and then extend the results to the directed case. Throughout this section, although treewidth is typically defined for undirected graphs, we will use treewidth of a directed graph to mean the trewewidth of the underlying undirected graph.

Undirected Graphs with Small Treewidth We begin by giving some intuition concerning our algorithmic strategy.

Our algorithm will actually compute the eccentricity of every node in the graph. Since diameter is the maximum eccentricity and radius the minimum eccentricity, we can compute these with only linear postprocessing.

Like many other algorithms, we make use of *portals*: the portals of a vertex subset A are those nodes of A that have edges going to outside A. Intuitively, finding a vertex subset A which has few portals allows us to divide the graph into relatively independent pieces. Specifically, if we compute single source shortest paths from all portals of A, we can augment the graph with weighted edges between portals to account for shortest paths that exit and re-enter A (or $V \setminus A$). Recursing on augmented graphs yields, for each node in A, the furthest node from it which is also in A (similarly for $V \setminus A$). If we could compute, for each node in A, the furthest node from it in $V \setminus A$ (and vice versa), we would be done.

But all of these paths pass through some portal. We know the distances from each node in A to each portal, and from each portal to each node in $V \setminus A$. We can think of the non-portals of A, the portals of A, and the nodes in $V \setminus A$ as forming a three-layered graph. We want to compute, for every node in the first layer, the furthest node in the third layer (using only two-hop paths). Note that the second layer only has as many nodes as there were portals.

As it turns out, this three-layered problem can be written as several max orthogonal range searching queries. To see this, consider a particular portal b in the middle layer. When is it the best portal to use to get to a node in the third layer? If a is a node in the first layer and c a node in the third, this happens when $d(a, b) + d(b, c) \leq d(a, b') + d(b', c)$ for every other portal b'. Using a standard inequality trick, we rearrange to get that $d(a, b) - d(a, b') \leq d(b', c) - d(b, c)$. If we think of each b' as a coordinate, we can use the right-hand side to transform each vertex c into a high-dimensional point. The set of c for which b is the best portal, given an a, are exactly those that fall into some orthogonal range. Furthermore, weighting each vertex by its distance from b allows us to recover the furthest one when we do a max query. Since these queries can be solved efficiently using a data structure of Chazelle [24], we can solve the three-layered problem efficiently:

THEOREM 3.1. ([24]) *Consider the range searching for maximum problem: we are given a set V of n points in d dimensions and a value function $v : V \to \mathcal{R}$. We want to answer queries of the form: given a range of the form $q = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d]$, what is $\max_{p \in q} v(p)$?*

On a word RAM, there is a data structure that solves this problem with $O(n \log^{d-1} n)$ preprocessing time, $O(n \log^{(d-1+\varepsilon)} n)$ space usage, and $O(\log^{d-1} n)$ query time.

Since the algorithm's running time is highly dependent on the number of portals, we use a result of Cabello and

Knauer [21] which finds a vertex subset with only k portals, but is unbalanced (one side may have k times as many nodes as the other):

LEMMA 3.1. ([21]) *Let $k \geq 1$ be a constant. Given a graph $G = (V, E)$ with $n > k + 1$ vertices and a tree decomposition of width at most k, we can find in $O(poly(k)n)$ time a subset of vertices $S \subseteq V$ such that S has between $\frac{n}{k+1}$ and $\frac{nk}{k+1}$ nodes and at most k portals.*

More formally, we call a directed graph $G = (V, E)$ a *three-layered graph* if there is a partition of V into A, B, C such that $E \subseteq A \times B \cup B \times C$, i.e. all edges go from A to B or from B to C. If G is a three-layered graph, we can also write G as (A, B, C, E). Using the orthogonal range searching data structure in Theorem 3.1, we are able to compute important distances in a three-layered graph. This serves as the key subroutine for solving diameter and radius on graphs of small treewidth:

THEOREM 3.2. *Suppose we have a weighted three-layered graph $G = (A, B, C, E)$. Furthermore, suppose that A and C have $O(n)$ nodes while B has only k nodes. Then we can compute $\max_{c \in C} \min_{b \in B} d(a, b) + d(b, c)$ for all $a \in A$ in $O(kn \log^{k-2} n)$ time.*

Proof. The key idea is as follows. Focus on some $b \in B$. We will preprocess all of the distances between B and C so that when given some $a \in A$, we can use its distances to the nodes of B to compute the subset of C whose shortest two-hop paths to a go through b. However, we don't actually compute this set; we instead use our orthogonal range searching data structure to return the *furthest* point in the set. This allows us to compute the furthest distance any node is from a, among nodes that use b as part of the shortest path. Looping over all $b \in B$ will then allow us to compute the desired quantity.

Fix $b \in B$. For each $c \in C$ and $b' \in B, b' \neq b$, we compute $d(b', c) - d(b, c)$. If we impose an ordering on B, this associates a $(k-1)$-dimensional vector with every $c \in C$. Suppose we have some $a \in A$ and $c \in C$ where b is the middle vertex in the shortest two-hop path from a to c. This means that $d(a, b) + d(b, c) \leq d(a, b') + d(b', c)$ for all other $b' \in B$. We can rewrite this as $d(a, b) - d(a, b') \leq d(b', c) - d(b, c)$ for all other $b' \in B$. In other words, given $a \in A$, the set of $c \in C$ for which b is the middle vertex with the shortest two-hop path from a to c are those c with vectors that fall in the axis-aligned box given by $d(a, b) - d(a, b')$.

However, by Theorem 3.1, there is a data structure that does this with only $O(n \log^{k-2})$ preprocessing time and $O(\log^{k-2} n)$ time per query. Note that the value function we use maps the point corresponding to c to $d(b, c)$. Hence the largest weight corresponds to the furthest point, and we can compute the distance from a to the furthest point by adding $d(a, b)$ to the weight returned.

We keep one data structure per $b \in B$, and now simply iterate over $a \in A$ and $b \in B$. For each $a \in A$, we select the furthest c over the two-hop distances computed. This takes $O(kn \log^{k-2} n)$ time.

We can now present the main algorithm:

THEOREM 3.3. *There is an algorithm that, given a tree decomposition of width at most k of an undirected weighted graph G, computes the eccentricity of every vertex in time $O(k^2 n \log^{k-1} n)$.*

Proof. By Lemma 3.1, we can find $S \subseteq V$ such that S has between $\frac{n}{k+1}$ and $\frac{nk}{k+1}$ vertices, at most k portals, and adding edges between portals of S does not change the treewidth of G. Finding S takes $O(poly(k)n)$ time.

We run Dijkstra from every portal of S. Since there are at most k portals, this takes $O(k^2 n + kn \log n)$ time. This yields the eccentricity of every portal. It remains to compute the eccentricity of non-portals of S and vertices in $V \setminus S$.

The eccentricity of a non-portal of S is either realized by a node in S or in $V \setminus S$. For the first case, we recurse on S augmented with weighted edges between portals corresponding to the distances between them that we computed via Dijkstra's. Any shortest path between nodes of S can be realized by taking a path in this graph; if it goes through at least two portals then our added portal-portal edge gives the correct distance. We note that Lemma 3.1 actually picks S such that its portals are a subset of a bag, and hence adding edges between portals does not alter the tree decomposition. Furthermore, we note that the algorithm can use the same tree decomposition when running on a subgraph with these additional portal-portal edges, since removing nodes only decreases the bag size (there will need to be additional preprocessing, but this is included in the $O(poly(k)n)$).

To cover the second case, we construct a three-layered graph where A consists of non-portal nodes of S, B consists of portals, and C is $V \setminus S$. We add edges from A to B and B to C weighted by the Dijkstra distances we computed for the portals. Any shortest path between a node in A and a node in C matches the cost of a two-hop path. Hence we can use Theorem 3.2 to compute, for each $a \in A$, the furthest $c \in C$.

Now, given $a \in A$, we have the furthest distance to any other node in S and the furthest distance to any node in $V \setminus S$. Hence we can compute the eccentricity of a (the max of these two).

Computing the eccentricities for every node of $V \setminus S$ is identical. We recurse on $V \setminus A$ augmented with the portals and weighted edges between portals. We also construct a three-layered graph where A is $V \setminus S$, B consists of portals, and C consists of non-portal nodes of S. We again invoke Theorem 3.2 on it, and take the max of the two computed furthest distances (for each node).

We now analyze the running time. Invoking Theorem 3.2 twice takes $O(kn\log^{k-2} n)$ time. Combining results and constructing graphs can be done in $O(k^2 + kn)$ time, which is dominated by $O(kn\log^{k-2} n)$.

We will stop recursing when we have k^3 nodes or fewer, which can be solved in $O(k^9)$ time by computing all-pairs shortest-paths. We guess that the algorithm runs in time $T'(n) = 4k(k+1)n\log^{k-1} n$, and we check this inductively. Notice that our case case is covered since $O(k^9)$ is dominated by $k^5\log^{k-1} k^3$.

Recall that $\frac{n}{k+1} \le |S| \le \frac{nk}{k+1}$, and that because of our base case, $\frac{0.5n}{k+1} \ge k$. The recurrence is
$T(n) \le kn\log^{k-2} n + T(|S|) + T(n - |S| + k)$.

$$\begin{aligned}
T(n) &\le kn\log^{k-2} n + T(|S|) + T(n-|S|+k)\\
&\le kn\log^{k-2} n + 4k(k+1)|S|\log^{k-1}|S|\\
&\quad + 4k(k+1)(n-|S|+k)\log^{k-1}(n-|S|+k)\\
&\le kn\log^{k-2} n + 4k(k+1)n\log^{k-1}\left(\frac{nk}{k+1}+k\right)\\
&\quad + 4k^2(k+1)\log^{k-1}\left(\frac{nk}{k+1}+k\right)\\
&\le kn\log^{k-2} n + 4k(k+1)n\log^{k-1}\left(\frac{n(k+0.5)}{k+1}\right)\\
&\quad + 4k^2(k+1)\log^{k-1}\left(\frac{n(k+0.5)}{k+1}\right)\\
&\le kn\log^{k-2} n\\
&\quad + 4k(k+1)n\log^{k-2} n(\log n - \log\frac{k+1}{k+0.5})\\
&\quad + 4k^2(k+1)\log^{k-1}\left(\frac{n(k+0.5)}{k+1}\right)\\
&\le kn\log^{k-2} n\\
&\quad + T'(n) - 4k(k+1)n\log^{k-2} n\frac{1}{2k+2}\\
&\quad + 4k^2(k+1)\log^{k-1}\frac{n(k+0.5)}{k+1}\\
&\le T'(n) - kn\log^{k-2} n + 4k^2(k+1)\log^{k-1} n
\end{aligned}$$

The negative term has at least as much magnitude as the positive term if $\frac{n}{\log n} \ge 4k(k+1)$, which is true because n is at least k^3. Hence our running time is indeed $O(k^2 n\log^{k-1} n)$. This completes the proof.

We can now use our eccentricities to compute the diameter and radius of a graph:

COROLLARY 3.1. *There are algorithms that, given a tree decomposition of width at most k of an undirected weighted graph G, compute* UNDIRECTEDDIAMETER *and* UNDIRECTEDRADIUS *in time $O(k^2 n\log^{k-1} n)$.*

Proof. We invoke Theorem 3.3, observing radius is the minimum eccentricity in the graph and diameter is the maximum eccentricity in the graph. We can recover both quantities in only $O(n)$ additional time.

By noticing that $g(k,n) = k^2 n\log^{k-1} n \le 2^{2k\log\log n} n$ can be upper-bounded by $2^{O(k\log k)} n^{1+o(1)}$ we prove Theorem 1.8 from the Introduction. This is because when $k \le \varepsilon\log n/\log\log n$ we can upper bound $g(n,k) = \tilde{O}(n^{1+\varepsilon})$ and otherwise $k > \varepsilon\log n/\log\log n$ and therefore $k^2 > \log n$ and $\log k > \log\log n/2$ and we can upper bound $g(n,k) = 2^{O(k\log k)}\cdot n$. Furthermore, if we use the Bodlaender et. al. 5-approximation tree decomposition algorithm, which runs in time $2^{O(k)}n$, the running time is $O(k^2 n\log^{5k-5} n)$, where k is now the treewidth of the input graph.

Directed Graphs With Small Treewidth We now explain simple modifications to Theorem 3.3 to compute the various directed eccentricities. As before, this means that we can compute diameter and radius, since they are simply the maximum and minimum eccentricities. A simple modification gives us max-eccentricities:

THEOREM 3.4. *There is an algorithm that, given a tree decomposition of width at most k of an directed weighted graph G, computes the max-eccentricity of every vertex in time $O(k^2 n\log^{k-1} n)$.*

Proof. We make a few modifications to the proof of Theorem 3.3. We must run forward and backward Dijkstra's from the portals of S (but this only doubles the running time). When recursing, we add directed edges between portals, weighted by the distance from the appropriate Dijkstra. We construct twice as many three-layered graphs; one weighted by forward distances from A to B and B to C and the other will have backward distances from A to B and B to C. The max-eccentricity of a node is just the maximum over its recursive value, the distance in the forward three-layered graph, and the distance in the backwards three-layered graph.

The running time analysis is identical.

COROLLARY 3.2. *There are algorithms that, given a tree decomposition of width at most k of an directed weighted graph G, compute* MAXDIAMETER *and* MAXRADIUS *in time $O(k^2 n\log^{k-1} n)$.*

Source-eccentricities are also easy:

THEOREM 3.5. *There is an algorithm that, given a tree decomposition of width at most k of an directed weighted graph G, computes the source-eccentricity of every vertex in time $O(k^2 n\log^{k-1} n)$.*

Proof. Again, we make modifications to the proof of Theorem 3.3. We run forward and backward Dijkstra's, and recurse with directed edges. We construct three-layered graphs weighted by forward distances bewteen A to B and B to C.

The running time analysis is identical.

COROLLARY 3.3. *There is an algorithm that, given a tree decomposition of width at most k of an directed weighted graph G, computes* SOURCERADIUS *in time* $O(k^2 n \log^{k-1} n)$.

It may be surprising that we can even solve MINDIAMETER and MINRADIUS efficiently, since they proved difficult in general graphs:

THEOREM 3.6. *There is an algorithm that, given a tree decomposition of width at most k of an directed weighted graph G, computes the min-eccentricity of every vertex in time $O(k^2 n \log^{2k-1} n)$.*

Proof. Again, we modify the proof of Theorem 3.3. We run forward and backward Dijkstra's, and recurse with directed edges. We construct three-layered graphs with twice as many nodes in the middle layer. One copy will have edges weighted by forward distances from A to B and B to C, while the other will have edges weighted by backward distances from A to B and B to C. Since distance meausres shortest paths, the furthest distance from any $a \in A$ will be the minimum of forward and backward distances to some node.

The running time analysis is almost identical, except invoking Theorem 3.2 now costs $O(kn \log^{2k-2} n)$ time. Hence we pay an additional $O(\log^k n)$ everywhere, to get a running time of $O(k^2 n \log^{2k-1} n)$.

COROLLARY 3.4. *There are algorithms that, given a tree decomposition of width at most k of an directed weighted graph G, compute* MINDIAMETER *and* MINRADIUS *in time* $O(k^2 n \log^{2k-1} n)$.

We need to do a little more work to get roundtrip-eccentricities. Since the paths of interest go through two portals, we end up with larger middle layers in our three-layered graph construction.

THEOREM 3.7. *There is an algorithm that, given a tree decomposition of width at most k of an directed weighted graph G, computes the roundtrip-eccentricity of every vertex in time $O(k^2 n \log^{k^2-1} n)$.*

Proof. We again modify the proof of Theorem 3.3. Run forward and backward Dijkstra's, and recurse with directed edges. We construct three-layered graphs with k^2 nodes in the middle layer, one per *pair* of portals. The weights from A to B will correspond to the sum of distance to the first portal and distance from the second portal, and weights from B to C will correspond to the sum of distance from the first portal and distance from the second portal.

Notice that two-hop paths in the three-layered graph now actually correspond to roundtrip distances between nodes in A and nodes in C, since these roundtrips must go through a portal each way.

The running time analysis is almost identical, except invoking Theorem 3.2 now costs $O(kn \log^{k^2-2} n)$ time. Hence we pay an additional $O(\log^{k^2-k} n)$ everywhere, to get a running time of $O(k^2 n \log^{k^2-1} n)$.

COROLLARY 3.5. *There are algorithms that, given a tree decomposition of width at most k of an directed weighted graph G, compute* ROUNDTRIPDIAMETER *and* ROUNDTRIPRADIUS *in time $O(k^2 n \log^{k^2-1} n)$.*

4 Conditional lower bounds

In this section we present our lower bound for Roundtrip Radius under the HS conjecture which is a good illustration of the constructions used in all our other reductions. All other lower bounds appear in the full version.

The HSE-Graph. All our reductions from HSE will start with the following simple representation of the HSE problem as a "radius-like" graph problem.

Given an instance A, B, U of HSE we create the following tripartite graph that we call an "HSE-graph" that we will utilize in our reductions. The vertex set is $A \cup B \cup U$ (we overload the notation slightly so that x denotes both a vertex and the corresponding subset in the original instance). The edge set E is as follows: for each $u \in U$ there is an edge to $x \in A \cup B$ if $u \in x$. The question becomes, is there a node $a \in A$ such that for all $b \in B$ there is a $u \in U$ such that $(a, u), (u, b) \in E$? Preprocess the HSE graph as follows. Suppose that there are some $a, a' \in A$ such that $N(a) \subseteq N(a')$ then we can remove a since if a is a hitting set, then so is a'. Now we can assume that for all $a, a' \in A$, there are $u, u' \in U$ such that $u \in N(a) \setminus N(a'), u' \in N(a') \setminus N(a)$. We will refer to this as the HSE-graph-problem.

LEMMA 4.1. *If for some $\varepsilon > 0$, there is an algorithm that can determine if a given directed, unweighted graph with n nodes and $m = O(n)$ edges has roundtrip radius 4 or 8 in $O(n^{2-\varepsilon})$ time, then the Hitting Set Conjecture is false.*

Proof. We will start from the HSE-graph G with partitions A', B', U and edge set E. We first build a gadget graph H from G as follows. H has vertex set $A \cup B \cup C \cup D$ where A is a copy of A', B is a copy of B' and C and D are copies of U. For $a \in A'$, let its copy in A also be a, and for $b \in B'$ let its copy in B also be b. For $u \in U$ let its copies in C and D be u_C and u_D, respectively.

If $a \in A', u \in U$, we create a directed 4-cycle connecting $a \in A$ and u_C and a directed 4-cycle connecting $a \in A$ and u_D as follows. If $(a, u) \in E$, then there is an edge from a to u_C and a path of length 3 directed from u_C to a where the internal nodes of the path are of degree 2 in H; additionally, there is an edge from u_D to a and a path of length 3 from a to u_D. If $(a, u) \notin E$, then the roles of the edges and 3-paths are reversed. That is, there is a 3-path from a to u_C and an edge from u_C to a and an edge from a to u_D and a 3-path from u_D to a. Call the set of internal nodes of all the 3-paths, X. Each edge (u, b) with $u \in U$, $b \in B$ is represented by two directed edges, $(u_C, b), (b, u_D)$. Note that any cycle in H has length at least 4 so that any roundtrip distance within H is also at least 4. Now, given H as a gadget, create two copies of H, H_1 on vertex partitions (A, B_1, C_1, D_1, X_1) and H_2 on (A, B_2, C_2, D_2, X_2) so that H_1 and H_2 are glued at A. Call this graph F and see Figure 1 for an illustration.

First suppose that the HSE-instance G was a "yes" instance, and there is some $a \in A$ such that for all $b \in B$, there is some $u \in U$ with $(a, u), (b, u) \in E$. Then we will show that a has roundtrip distance at most 4 to all nodes in F and hence the roundtrip radius is at most 4. To see this, first note that by construction, a is on a cycle of length 4 to every node of $D_1 \cup C_1 \cup D_2 \cup C_2$. For any other node $a' \in A$, let $u, u' \in U$ be nodes such that $(a, u), (a', u') \in E, (a, u'), (a', u) \notin E$ (recall such u, u' exist). Then $a \to u_{C_1} \to a' \to u_{D_1} \to a$ is a directed 4-cycle in F. Finally, for any $b_i \in B_i$ for $i = 1, 2$, if $u \in U$ is such that $(a, u), (u, b) \in G$, the following is a directed 4-cycle in F: $a \to u_{C_i} \to b_i \to u_{D_i} \to a$.

Now suppose that the roundtrip radius of F is < 8 and we will show that the original graph G must be a "yes" instance. We first claim that no node of $F \setminus A$ can be a center.

Case 1. Suppose that some node u_{C_1} is a center (the cases $u_{C_2}, u_{D_1}, u_{D_2}$ are symmetric). Then consider the roundtrip shortest path to u_{C_2}. Either the portion of the path from u_{C_1} to u_{C_2}, or the one from u_{C_2} to u_{C_1}, must have length at most 3. Assume, w.l.o.g. that $d(u_{C_1} \to u_{C_2}) \leq 3$, and note that the path must go through A. None of the 3-paths can be used, since the length would become > 3, which implies that it must be of the form $u_{C_1} \to a \to u_{C_2}$ for some $a \to A$. However, by construction, if $(a, u_{C_2}) \in E(F)$ then $(a, u) \in E(G)$ and $(u_{C_1}, a) \notin E(F)$. Hence u_{C_1} cannot be a center.

Case 2. Suppose that some node x in X is the center and let u be the closest node in $C_1 \cup C_2 \cup D_1 \cup D_2$ to x. Note that any roundtrip path from x must go through u, which implies that u can only be a better center than x. But by case 1, u cannot be the center and therefore neither can x.

Case 3. Now consider any two nodes $b_1 \in B_1, b_2 \in B_2$. By construction, $d(b_1, b_2), d(b_2, b_1) \geq 4$ and hence the roundtrip distance is at least 8. Hence no node of $B_1 \cup B_2$ can be a center.

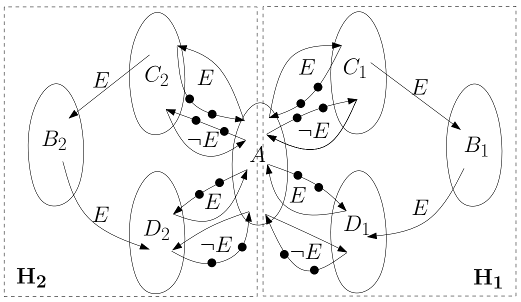

Figure 1: The reduction from HSE to Roundtrip Radius.

Hence the center of F is some node $a \in A$. Consider the roundtrip distance from a to any $b_1 \in B_1$. It is supposed to be at most 7. Any path from a to b_1 that does not go directly from a to some node of C_1 to b_1 must have length at least 4. Similarly, any path from b_1 to a that does not go directly from a to some node of D_1 to a must have length at least 4. Thus, if the roundtrip radius is < 8, one of the pieces of the roundtrip path (from a to b_1 and from b_1 to a) must be of length 2, as otherwise the roundtrip path would be of length at least 8. Hence there is some $u \in U$ for which $(a, u), (u, b) \in E$ and the original graph G is a "yes" instance of HSE.

To complete the proof, note that our new graph F has $O(n|U|)$ nodes and $O(n|U|)$ edges. This implies that a subquadratic algorithm for sparse graphs that distinguished between roundtrip radius 4 and 8 will solve the HSE problem in $O(n^{2-\varepsilon} \cdot |U|^{2-\varepsilon})$ time, for some $\varepsilon > 0$, which refutes the HS conjecture.

Finally, we observe that the treewidth (in fact, pathwidth) of the graph in our construction is $O(|U|)$ since by removing all nodes in the $C \cup D$ parts of the graph we are left with a disconnected set of paths. Thus, an algorithm that can compute Radius on treewidth (or pathwidth) k graphs in $2^{o(k)} \cdot n^{2-\varepsilon}$ can be used to solve the HSE problem where $|U| = \omega(\log n)$ in $O(n^{2-\varepsilon})$ time, refuting the HS conjecture.

References

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of lcs and other sequence similarity measures. *FOCS*, 2015.

[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.

Radius Variants			
Problem	Definition	Upper Bound	HS Conjecture
UNDIRECTEDRADIUS	$\min_{v^*} \max_v d(v^*, v)$	3/2 in $\tilde{O}(m\sqrt{n})$ [[47]]	3/2 [Thm 1.1]
SOURCERADIUS	$\min_{v^*} \max_v d(v^*, v)$	2 in $\tilde{O}(m\sqrt{n}\log M)$ [Thm 2.1]	2 [Thm 1.4]
MAXRADIUS	$\min_{v^*} \max_v \max\{d(v^*, v), d(v, v^*)\}$	2 in $\tilde{O}(m)$ [metric]	2
MINRADIUS	$\min_{v^*} \max_v \min\{d(v^*, v), d(v, v^*)\}$	n	2
MINRADIUS on DAGs	$\min_{v^*} \max_v \min\{d(v^*, v), d(v, v^*)\}$	3 in $\tilde{O}(m\sqrt{n}\log M)$ [Thm 2.3]	2
ROUNDTRIPRADIUS	$\min_{v^*} \max_v \{d(v^*, v) + d(v, v^*)\}$	2 in $\tilde{O}(m)$ [metric]	2 [Thm 1.5]

Table 1: Our Bounds for Various Radius Problems. Missing proofs are given in the full version.

Diameter Variants			
Problem	Definition	Upper Bound	OV Conjecture
UNDIRECTEDDIAMETER	$\max_{u,v} d(u, v)$	3/2 in $\tilde{O}(m\sqrt{n})$ [47]	3/2 [[47]]
MAXDIAMETER	$\max_{u,v} d(u \to v)$	3/2 in $\tilde{O}(m\sqrt{n})$ [47]	3/2 [[47]]
MINDIAMETER	$\max_{u,v} \min\{d(u \to v), d(v \to u)\}$	n^ϵ in $\tilde{O}(mn^{1-\epsilon})$ [Lem 2.1]	2 on weighted
MINDIAMETER on DAGs	$\max_{u<v} d(u \to v)$	2 in $\tilde{O}(m)$ [Thm 2.2]	3/2
ROUNDTRIPDIAMETER	$\max_{u,v}\{d(u \to v) + d(v \to u)\}$	2 in $\tilde{O}(m)$ [metric]	3/2

Table 2: Our Bounds for Various Diameter Problems. Missing proofs are given in the full version.

[3] Amir Abboud, Richard Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 218–230, 2015.

[4] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014.

[5] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *ICALP (1)*, pages 39–51, 2014.

[6] 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.

[7] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17:209–223, 1997.

[8] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58, 2015.

[9] Alex Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, pages 725–730, 1950.

[10] Murray A Beauchamp. An improved index of centrality. *Behavioral Science*, 10(2):161–163, 1965.

[11] 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.

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

[13] Hans L Bodlaender. *Dynamic programming on graphs with bounded treewidth*. Springer, 1988.

[14] Hans L Bodlaender. Treewidth: characterizations, applications, and computations. In *Graph-theoretic concepts in computer science*, pages 1–14. Springer, 2006.

[15] Hans L Bodlaender, Pål Grønås Drange, Markus S Dregi, Fedor V Fomin, Daniel Lokshtanov, and Michal Pilipczuk. An o (cˆ kn) 5-approximation algorithm for treewidth. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 499–508. IEEE, 2013.

[16] Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the square - on the complexity of quadratic-time solvable problems. *CoRR*, abs/1407.4972, 2014.

[17] 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*, 2015. accepted.

[18] Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 661–670, 2014.

[19] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. *FOCS*, 2015.

[20] Karl Bringmann and Wolfgang Mulzer. Approximability of the Discrete Fréchet Distance. In *31st International*

Symposium on Computational Geometry (SoCG 2015), pages 739–753, 2015.

[21] Sergio Cabello and Christian Knauer. Algorithms for graphs of bounded treewidth via orthogonal range searching. *Computational Geometry*, 42(9):815–824, 2009.

[22] Marco Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mikhailin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:148, 2015.

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

[24] B. Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *J. ACM*, 47(6):1028–1047, 2000.

[25] S. Chechik, D. Larkin, L. Roditty, G. Schoenebeck, R. E. Tarjan, and V. Vassilevska Williams. Better approximation algorithms for the graph diameter. In *Proc. SODA*, 2014.

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

[27] 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.

[28] F. R. K. Chung. Diameters of graphs: Old problems and new results. *Congr. Numer.*, 60:295–317, 1987.

[29] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. Computing classic closeness centrality, at scale. *CoRR*, abs/1409.0035, 2014.

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

[31] L. Cowen and C. Wagner. Compact roundtrip routing for digraphs. In *SODA*, pages 885–886, 1999.

[32] Lenore J Cowen and Christopher G Wagner. Compact roundtrip routing in directed networks. In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pages 51–59. ACM, 2000.

[33] 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.

[34] Rod G Downey and Michael Ralph Fellows. *Parameterized complexity*, volume 3. springer Heidelberg, 1999.

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

[36] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms and Applications*, 3(3):1–27, 1999.

[37] Jörg Flum and Martin Grohe. Parameterized complexity theory, volume xiv of texts in theoretical computer science. an eatcs series, 2006.

[38] 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.

[39] Archontia C. Giannopoulou, George B. Mertzios, and Rolf Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *CoRR*, abs/1506.01652, 2015.

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

[41] R. Impagliazzo and R. Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[42] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[43] Piotr Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 428–434. ACM, 1999.

[44] R. Niedermeier. Invitation to fixed-parameter algorithms. pages 84–103, 2004.

[45] M. Pătraşcu and L. Roditty. Distance oracles beyond the thorup–zwick bound. In *Proc. FOCS*, pages 815–823, 2010.

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

[47] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 515–524, New York, NY, USA, 2013. ACM.

[48] Liam Roditty, Mikkel Thorup, and Uri Zwick. Roundtrip spanners and roundtrip routing in directed graphs. *ACM Transactions on Algorithms*, 4(3), 2008.

[49] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. In *ESA*, pages 580–591, 2004.

[50] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

[51] Barna Saha. Faster language edit distance, connection to all-pairs shortest paths and related problems. In *FOCS*, 2015.

[52] Barbaros C Tansel, Richard L Francis, and Timothy J Lowe. State of the art - location on networks: a survey. part i: the p-center and p-median problems. *Management Science*, 29(4):482–497, 1983.

[53] Mikkel Thorup. Quick k-median, k-center, and facility location for sparse graphs. *SIAM Journal on Computing*, 34(2):405–432, 2005.

[54] O. Weimann and R. Yuster. Approximating the diameter of planar graphs in near linear time. In *Proc. ICALP*, 2013.

[55] R. Williams. A new algorithm for optimal constraint satisfaction and its implications. In *Proc. ICALP*, pages 1227–1237, 2004.

[56] V. Vassilevska Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proc. FOCS*, pages 645–654, 2010.

[57] C. Wulff-Nilsen. Wiener index, diameter, and stretch factor of a weighted planar graph in subquadratic time. *Technical report, University of Copenhagen*, 2008.

[58] Raphael Yuster. Computing the diameter polynomially faster than apsp. *arXiv preprint arXiv:1011.6181*, 2010.