Algorithms and Hardness for Diameter in Dynamic Graphs

Bertie Ancona
MIT, Cambridge, MA, USA
bancona@mit.edu

Monika Henzinger
University of Vienna, Austria
monika.henzinger@univie.ac.at

Liam Roditty
Bar Ilan University, Ramat Gan, Israel
liam.roditty@biu.ac.il

Virginia Vassilevska Williams
MIT, Cambridge, MA, USA
virgi@mit.edu

Nicole Wein
MIT, Cambridge, MA, USA
nwein@mit.edu

Abstract

The diameter, radius and eccentricities are natural graph parameters. While these problems have been studied extensively, there are no known *dynamic* algorithms for them beyond the ones that follow from trivial recomputation after each update or from solving dynamic All-Pairs Shortest Paths (APSP), which is very computationally intensive. This is the situation for dynamic *approximation* algorithms as well, and even if only edge insertions or edge deletions need to be supported.

This paper provides a comprehensive study of the dynamic approximation of Diameter, Radius and Eccentricities, providing both conditional lower bounds, and new algorithms whose bounds are optimal under popular hypotheses in fine-grained complexity. Some of the highlights include:

- Under popular hardness hypotheses, there can be no significantly better *fully dynamic approximation* algorithms than recomputing the answer after each update, or maintaining full APSP.
- Nearly optimal *partially dynamic* (incremental/decremental) algorithms can be achieved via efficient reductions to (incremental/decremental) maintenance of Single-Source Shortest Paths. For instance, a nearly $(3/2+\epsilon)$-approximation to Diameter in directed or undirected n-vertex, m-edge graphs can be maintained decrementally in total time $m^{1+o(1)}\sqrt{n}/\epsilon^2$. This nearly matches the static 3/2-approximation algorithm for the problem that is known to be conditionally optimal.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases fine-grained complexity, graph algorithms, dynamic algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.13

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1811.12527`.

Funding *Monika Henzinger*: The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement No. 340506.
Virginia Vassilevska Williams: Supported by an NSF CAREER Award, NSF Grants CCF-1417238, CCF-1528078 and CCF-1514339, a BSF Grant BSF:2012338 and a Sloan Research Fellowship.
Nicole Wein: Supported by an NSF Graduate Fellowship and NSF Grant CCF-1514339.

Acknowledgements The authors would like to thank Roei Tov for discussions.

1 Introduction

Computing the shortest paths distances between all pairs of vertices in a graph, the All-Pairs Shortest Paths (APSP) problem, has been studied since the beginning of computer science as a field. Nevertheless, the fastest known algorithms [26, 20, 19] for APSP in n-vertex m-edge graphs are only slightly faster (by $n^{o(1)}$ factors) than the simple $\tilde{O}(mn)$ time[1] algorithm running Dijkstra's algorithm from every vertex[2]. For dense graphs with small integer weights there are improved algorithms [22, 28] using fast matrix multiplication [23, 17], but these algorithms are not faster than mn for sparser graphs or when the weights can be large.

There are many important graph parameters that can be easily computed when all the distances are known. These include the *eccentricity* of each vertex (the maximum length of a shortest path from the vertex to another vertex), the graph *diameter* (the maximum over all eccentricities), the *radius* (the minimum over all eccentricities) and many more. These parameters are of particular importance in the analysis of social networks (e.g. [3]), but also in graphs generated for entities such as images and search queries (and web pages).

Unfortunately, there are no significantly faster algorithms to compute these parameters than just solving APSP, and this is far from practical. In many cases the analyzed networks are so large that even enumerating all pairs of vertices is prohibitively expensive. Thus, obtaining all pairwise distances is essentially impossible. For graph parameters, on the other hand, the output is a single number; in principle looking at all vertex pairs might not be necessary, and subquadratic time algorithms (in the number of vertices) might exist for sparse graphs (whereas quadratic time is necessary for APSP as this is the size of the output). The existence of such fast algorithms is an important, practically motivated question.

In recent years, much progress has been made in understanding the complexity of graph parameter computation. Results from fine-grained complexity give that even obtaining a $(3/2-\epsilon)$-approximation for graph diameter [21] or radius [2], or a $(5/3-\epsilon)$-approximation of all eccentricities [8] (for $\epsilon > 0$) requires $n^{2-o(1)}$ time even in very sparse graphs, assuming the Strong Exponential Time Hypothesis (SETH) and related conjectures. Even stronger hardness results were obtained by Backurs et al. [6], altogether showing that most of the known algorithms for diameter, radius and eccentricities are conditionally optimal.

In addition to computing graph parameters in a static graph, a very natural goal is to maintain estimates of these parameters in a *dynamic* graph, where edges are inserted and deleted. In this setting, we would like to have a fast algorithm which preprocesses the given graph, and builds a data structure which can support edge updates efficiently and can answer queries about the parameter of interest in the current state of the data structure. This dynamic version of the problems is even more practically motivated, as real networks are naturally dynamic.

Unfortunately, the state of the art of dynamic algorithms for graph parameters such as Diameter is somewhat disappointing. The best known dynamic algorithms either just use the best known dynamic algorithms for APSP, or recompute the parameter estimate from scratch after each update. This leads to the following bounds:

(1) Demetrescu and Italiano [10] obtained a fully dynamic exact APSP algorithm with an amortized update time of $\tilde{O}(n^2)$ and $O(1)$ query time; this is the best exact dynamic algorithm for the graph parameters as well. Abboud and Vassilevska W. [1] showed that under SETH, any $(4/3-\epsilon)$-approximation fully dynamic algorithm for diameter (for $\epsilon > 0$) requires

1 $\tilde{O}$ notation supresses polylogarithmic factors.
2 after Johnson's trick to make the weights nonnegative

$n^{2-o(1)}$ amortized update or query time even in sparse graphs. Thus the APSP approach is conditionally optimal for fully dynamic $(4/3-\epsilon)$-approximate diameter algorithms. It is unclear however whether a 4/3-approximation with better update time is possible, and whether the APSP bounds are best for Radius.

(2) By recomputing the parameter estimates after each query, one can maintain a 2-approximation for Diameter in directed graphs and Radius in undirected graphs in worst case $O(m+n)$ time per update, and a $(2+\epsilon)$-approximation for all Eccentricities in directed graphs for all ϵ in $\tilde{O}(m/\epsilon)$ time per update using the algorithm of Backurs et al. [6]. One can also maintain a 3/2-approximation for Diameter and Radius, and a 5/3-approximation of all Eccentricities in worst case time $\tilde{O}(m^{3/2})$ per update using the algorithms of [21, 8] . More algorithms follow from the static results of Cairo, Grossi and Rizzi [7]. Can any of these algorithms be improved or are they conditionally optimal? The only related lower bounds here are (a) by Henzinger et al. [14] which showed that under the Online Matrix Vector hypothesis (OMv), any fully dynamic Diameter algorithm that achieves a $(2-\epsilon)$-approximation for undirected *weighted* graphs, or any finite approximation in directed graphs needs $n^{0.5-o(1)}$ amortized update time and (b) by Henzinger et al. [15] which proved under the combinatorial Boolean Matrix Multiplication conjecture any fully dynamic Diameter or Eccentricity algorithm that achieves a $(4/3-\epsilon)$-approximation in undirected *unweighted* graphs with $n^{3-o(1)}$ preprocessing time requires $n^{2-o(1)}$ update or query time (and the same result for undirected *weighted* graph using the APSP conjecture). While these results give some limitation, they are far from tight.

The first contributions of our paper are strong conditional lower bounds for fully dynamic graph parameter estimation. Our first result is a strengthening of the conditional lower bound for Diameter of [1]: we increase the approximation ratio from $(4/3-\epsilon)$ to $(3/2-\epsilon)$.

▶ **Theorem 1.** *Under SETH, every fully dynamic $(3/2-\epsilon)$-approximation algorithm for Diameter with polynomial preprocessing time requires $n^{2-o(1)}$ amortized update or query time in the word-RAM model of computation with $O(\log n)$ bit words, even for dynamic undirected unweighted graphs that are always sparse.*

The same limitation applies for fully dynamic $(5/3-\epsilon)$-approximation algorithms for Eccentricities with polynomial preprocessing time, and for fully dynamic $(3/2-\epsilon)$-approximation algorithms for Radius with polynomial preprocessing time, under the related Hitting Set hypothesis.

These conditional lower bounds imply that the $\tilde{O}(m^{3/2})$ time estimation algorithms that recompute the answer from scratch are optimal in the sense that any improvement of the approximation factor causes the update time to grow to n^2, and Demetrescu and Italiano's algorithm achieves $\tilde{O}(n^2)$ update time even for the exact maintenance of APSP.

We also show that recomputing a 2-approximation from scratch in linear time is close to optimal under SETH.

▶ **Theorem 2.** *Under SETH, any fully dynamic algorithm with polynomial preprocessing time that can maintain for $\epsilon > 0$ any of the following in an n node, m-edge undirected unweighted graph requires either $m^{1-o(1)}$ amortized update or query time, even when $m = \tilde{O}(n)$:*

- *a $(2-\epsilon)$-approximation of the eccentricity of a fixed vertex, or*
- *a $(2-\epsilon)$-approximation of the Radius, or*
- *a $(2-\epsilon)$-approximation of the Diameter.*

This result significantly strengthens the OMv based lower bound of [14]: the update time lower bound is now linear as opposed to $\sqrt{n}$, the result holds for unweighted graphs as well, it also holds for Radius and single-node eccentricity, and it has implications for partially dynamic algorithms with worst case time bounds, unlike the one of [14]. Furthermore, the lower bound for the eccentricity of a fixed vertex is tight in the sense that one can simply recompute the answer exactly from scratch in time $O(m)$.

Much stronger lower bounds are possible for *directed* graphs. Our first hardness result for directed graphs is that under SETH and the Hitting Set hypothesis, respectively, nearly quadratic time is needed for Eccentricities and Radius, even for $(2-\epsilon)$-approximation. (Recall that for undirected graphs we could only show this for $(3/2-\epsilon)$-approximate Radius and $(5/3-\epsilon)$-approximate Eccentricities.) This means that for directed graphs, recomputing a 2-approximation from scratch (in linear time) after each update is very much optimal – for any better approximation one might as well use the exact dynamic APSP algorithms.

▶ **Theorem 3.** *Every fully dynamic algorithm with polynomial preprocessing time for $(2-\epsilon)$-approximate (for $\epsilon > 0$) Eccentricities (under SETH) or Radius (under a version of the Hitting Set Hypothesis) in directed, unweighted graphs with n vertices and $m = \tilde{O}(n)$ edges requires amortized update or query time $m^{2-o(1)}$.*

Surprisingly, we also show conditionally that no *finite* approximation can be maintained in *sublinear* time. Henzinger et al. [14], building upon [1], showed that any finite approximation for Diameter in directed graphs requires $m^{0.5-o(1)}$ time under the OMv Hypothesis. We strengthen the lower bound to linear, using a very natural hypothesis on the complexity of k-Cycle.

All known algorithms for detecting k-cycles in sparse directed graphs with m edges run at best in time $m^{2-c/k}$ for various small constants c [27, 4, 9], even if you use powerful tools such as fast matrix multiplication. A natural hypothesis completely consistent with the state of the art of cycle detection is that one needs $m^{2-f(k)-o(1)}$ time to find a k-cycle, for some continuous (over the reals) $f(k)$ that goes to 0 as k goes to infinity. Let us call this the k-Cycle Hypothesis. We obtain:

▶ **Theorem 4.** *Under the k-Cycle Hypothesis, any fully dynamic algorithm with polynomial preprocessing time that can maintain a* finite *approximation for any of the following in an n node, $m = \tilde{O}(n)$-edge directed unweighted graph requires either $m^{1-o(1)}$ amortized update or query time:*
- *the eccentricity of a fixed vertex, or*
- *the Radius, or*
- *the Diameter.*

All known approaches to estimating graph proximity parameters such as the Diameter, at the very least require maintaining approximate distances from a single node, up to some distance. The conditional lower bounds above say that even if we only want to maintain an estimate of the largest distance from a fixed node, and even if that distance is never more than a constant, we still need linear update time. Thus, to have better than 2 approximations of our undirected distance parameters or any finite approximation in the directed case that can be maintained in sublinear time we probably need to abandon our need for fully dynamic algorithms. We thus turn to *partially* dynamic algorithms that handle either only edge insertions (incremental) or only edge deletions (decremental).

Our conditional lower bounds for the fully dynamic setting also apply to incremental and decremental algorithms that have *worst case* update and query time guarantees. This is due to the nature of our reductions: they all produce an initial graph on which we perform update stages that only insert or only delete (we can choose which) a small batch of edges, ask a query and undo the changes just made, returning to the initial graph. An incremental/decremental algorithm can be used to implement such reductions by performing the deletions/insertions by rolling back the data structure. Because of this, we have very strong worst case lower bounds, and it makes sense to focus on amortized guarantees instead.

The strong conditional lower bounds in the static case, imply strong limitations for partially dynamic algorithms as well. For *undirected* graphs, these limitations are as follows: Due to [21, 6, 8], under SETH, every incremental and decremental algorithm for Diameter in n node undirected unweighted graphs requires total time at least $m^{3/2-o(1)}$ to maintain a $(8/5-\epsilon)$-approximation, and at least $m^{2-o(1)}$ total time to maintain a $(3/2-\epsilon)$-approximation for $\epsilon > 0$ under $m = \tilde{O}(n)$ insertions or deletions. For Eccentricities, the static conditional lower bounds are slightly stronger. For partially dynamic algorithms they imply that under SETH, for every $k \geq 1$, maintaining a $((3k+2)/(k+2)-\epsilon)$-approximation for $\epsilon > 0$ requires total time $m^{1+1/k}$ for $m = \tilde{O}(n)$ insertions or deletions. For Radius, they just imply that under the Hitting Set hypothesis [24, 2] maintaining a $(3/2-\epsilon)$-approximation requires total time $m^{2-o(1)}$ even in a sparse graph.

For *directed* graphs, there are stronger lower bounds: maintaining a $(2-\epsilon)$-approximation for Radius and Eccentricities requires total time $m^{2-o(1)}$ under Hitting Set and SETH, respectively [2, 6].

The incremental lower bounds directly follow from the static ones by starting from an empty graph and inserting edges until we reach the graph from the static construction. The decremental lower bounds hold since the static lower bound instances are all subgraphs of the same global graph, independent of the SAT/Hitting Set instance that the reduction is trying to solve; thus we start with the global graph and delete edges until reaching the graph from the static construction.

A natural question is, are these conditional lower bounds tight? Can one create partially dynamic algorithms that can achieve the same total runtime as the known static approximation algorithms? We give positive answers to these questions by developing new partially dynamic algorithms that are essentially optimal. Our algorithms are actually very efficient reductions to incremental and decremental single source shortest paths (SSSP), so that any improvement over dynamic SSSP would improve our parameter estimation algorithms.

Let D_0 and D_f be the initial and final values of the diameter, respectively. Let $T_{inc}(n, m, k, \epsilon)$ (resp., $T_{dec}(n, m, k, \epsilon)$) be the total time of an incremental (decremental) approximate SSSP algorithm from source u that maintains an estimate $d'(u, v)$ for all v such that if $d(u, v) \leq k$ then $(1-\epsilon)d(u, v) \leq d'(u, v) \leq d(u, v)$. For directed graphs we assume that the approximate SSSP algorithm works in directed graphs, and for undirected graphs, the SSSP algorithm only needs to work in undirected graphs. Our black-box reductions can be summarized in the theorem below.

▶ **Theorem 5.** *There is a Las Vegas randomized algorithm for incremental (resp., decremental) diameter in unweighted, directed graphs against an oblivious (resp., adaptive) adversary that given $\epsilon > 0$, runs in total time $\tilde{O}(\max_{D_f \leq D' \leq D_0}\{T_{inc}(n, m, D', \epsilon)\frac{\sqrt{n/D'}}{\epsilon^2}\})$ (resp., $\tilde{O}(\max_{D_0 \leq D' \leq D_f}\{T_{dec}(n, m, D', \epsilon)\frac{\sqrt{n/D'}}{\epsilon^2}\})$) with high probability, and maintains an estimate $\hat{D}$ such that $\frac{2(1-\epsilon)}{3}D - \frac{2}{3} \leq \hat{D} \leq D$ where D is the diameter of the current graph.*

We obtain similar black-box reductions for nearly $(5/3+\epsilon)$-approximate Eccentricities and $(3/2+\epsilon)$-approximate Radius in undirected graphs.

Henzinger et al. [13] obtained a randomized $(1+\epsilon)$-approximate decremental algorithm for SSSP in undirected unweighted graphs against an oblivious adversary with total expected update time $m^{1+o(1/\epsilon)}$. As an immediate corollary we obtain:

▶ **Corollary 6.** *There is a Las Vegas randomized algorithm for decremental diameter in unweighted, undirected graphs against an oblivious adversary that given $\epsilon > 0$, runs in total time $m^{1+o(1/\epsilon)}\sqrt{n}/\epsilon^2$ in expectation, and maintains an estimate $\frac{2(1-\epsilon)}{3}D - \frac{2}{3} \leq \hat{D} \leq D$, where D is the diameter of the current graph.*

Due to lower bounds in the static setting described above, this result is conditionally optimal in terms of both running time and approximation factor except for a small loss in the approximation factor. A similar result hold for decremental undirected Radius with a conditionally essentially optimal approximation factor. A similar result also holds for Eccentricities, which is conditionally essentially optimal in terms of both running time and approximation factor.

For decremental algorithms in directed graphs and for incremental algorithms in undirected or directed graphs, the best known algorithms for SSSP up to distance k are achieved by the Even and Shiloach Trees data structure [11], giving amortized update time $O(k)$. Henzinger and King recognized that this data structure can be extended to directed graphs [12]. As a corollary we obtain:

▶ **Corollary 7.** *There is a Las Vegas randomized algorithm for incremental/decremental diameter in unweighted, directed graphs that given $\epsilon > 0$, runs in total time $\tilde{O}(m\sqrt{nD_{\max}}/\epsilon^2)$ with high probability where $D_{\max}$ is the maximum diameter throughout the algorithm, and maintains an estimate $\hat{D}$ such that $\frac{2(1-\epsilon)}{3}D - 1 \leq \hat{D} \leq D$, where D is the diameter of the current graph. The incremental algorithm works against an oblivious adversary and the decremental algorithm works against an adaptive adversary.*

Similar results hold for Radius and Eccentricities but only for undirected graphs. Recall that static conditional lower bounds rule out such algorithms in directed graphs.

The algorithms so far are all randomized. We present some deterministic incremental algorithms as well, again via a reduction to incremental SSSP. Let D_0, D_f and $T_{inc}(n, m, k, \epsilon)$ be as before.

▶ **Theorem 8.** *There is a deterministic algorithm for incremental diameter in unweighted, directed graphs that, for any ϵ with $0 < \epsilon < 2$, runs in total time $\tilde{O}(\max_{D_f \leq D' \leq D_0}\{(T_{inc}(n, m, D', \epsilon) + m)n/(\epsilon^2 D')\})$, and maintains an estimate $\hat{D}$ such that $(1-\epsilon)D \leq \hat{D} \leq D$, where D is the diameter of the current graph.*

Using Even and Shiloach trees we obtain as a corollary a deterministic incremental $(1+\epsilon)$-approximation algorithm for diameter with total update time $\tilde{O}(mn/\epsilon^2)$. The running time is essentially tight for $\epsilon < 1/2$ by the SETH based quadratic lower bound for $(3/2-\delta)$-approximate static diameter [21]. Similar algorithms with essentially tight running times hold for radius in directed graphs and eccentricities in directed, strongly connected graphs.

1.1 Our techniques for partially dynamic algorithms

Our partially dynamic nearly 3/2-approximation algorithms for diameter and radius and our nearly 5/3-approximation algorithm for eccentricities are based on known algorithms in the static setting [21, 8]. These static algorithms work by carefully choosing a set U of vertices, performing SSSP from every vertex in U, and showing that at least one of these SSSP instantiations yields a good estimate for the parameter of interest. The set U is chosen as follows. We pick a random sample S of $\tilde{\Theta}(\sqrt{n})$ vertices and let w^* be the vertex that is farthest from S; that is, w^* is the vertex that maximizes $\min_{s \in S} d(w^*, s)$. Then, letting $N(w, \sqrt{n})$ be the closest $\sqrt{n}$ vertices to w, we set $U = S \cup \{w^*\} \cup N(w^*, \sqrt{n})$.

Adapting these static algorithms to the dynamic setting presents two main challenges:

(1) Firstly, given a set S of vertices, the farthest vertex w^* from S can change over time. We wish to minimize the total number of vertices that we ever run dynamic SSSP from, as reinitializing dynamic SSSP from a new vertex is expensive. Suppose we run dynamic SSSP from every vertex in $N(w^*, \sqrt{n})$ at all times. Then, every time w^* changes, we must reinitialize the dynamic SSSP data structure from $\sqrt{n}$ new vertices. If w^* changes frequently, this is prohibitively slow. To overcome this issue, we show that it suffices to choose a vertex w that *approximates* w^* (for a careful notion of approximation); and furthermore, by doing so we can limit the number of times we choose a new w.

Due to inherent differences between the incremental and decremental settings, we choose w in different ways in the different settings. In the decremental setting, distances can only increase, so our current choice of w can only become a poor approximation for w^* if $d(w^*, S)$ increases. Then, we use the fact that $d(w^*, S)$ is monotonically increasing to bound the number of times we need to choose a new w.

The incremental setting is more involved. Since distances can only decrease, our current choice of w becomes a poor approximation of w^* if $d(w, S)$ decreases. A challenge arises because unlike $d(w^*, S)$, the distance $d(w, S)$ does not change monotonically. One can imagine a scenario in which whenever we choose a new w, an edge is added causing $d(w, S)$ to immediately decrease to 1, which mandates that we choose a new w. We address this challenge by carefully employing randomness against an oblivious adversary. We argue that by randomly sampling w from a specifically chosen set of vertices, in expectation it will take a long time for w to become a poor approximation for w^*.

(2) Secondly, we wish to apply a partially dynamic SSSP algorithm as a subroutine, however the state of such algorithms is much better for undirected graphs than directed graphs. For instance, for *undirected* decremental graphs, there is a randomized $(1+\epsilon)$-approximate SSSP algorithm that runs amortized $m^{o(1)}$ time [13] (and it is believed, but not published, than a similar result is possible for incremental graphs), while for incremental/decremental *directed* graphs the best known algorithms for SSSP up to distance k run in amortized time $O(k)$ [11]. To address this discrepancy, we carefully exploit the fact that longer paths are easier to hit by randomly sampling: we augment the algorithm with an additional subsampling routine that quadratically decreases the time dependence on the diameter D.

2 Preliminaries

Let $G = (V, E)$ be a graph, where $|V| = n$ and $|E| = m$. For every $u, v \in V$ let $d_G(u, v)$ be the length of the shortest path from u to v. We omit the subscript when G is clear from context. Let $N_{out}(v, s)$ (resp., $N_{in}(v, s)$) be the set of the s closest outgoing (incoming) vertices of v, where ties are broken by taking the vertex with the smaller ID. The eccentricity $\varepsilon(v)$ of a vertex v is defined as $\max_{u \in V} d(v, u)$. The diameter D of a graph is $\max_{v \in V} \varepsilon(v)$. The radius R of a graph is $\min_{v \in V} \varepsilon(v)$.

2.1 Algorithms

For all of our algorithms for diameter, radius, and eccentricities in undirected graphs as well as diameter in directed graphs, we assume that the diameter is finite. One can easily check if this is the case by running a dynamic reachability algorithm from a single vertex. For our partially dynamic algorithms, we let D_0 and D_f be the initial and final values of the diameter, respectively. Similarly, R_0 and R_f are the initial and final values of the radius, respectively.

The running times of our randomized algorithms are with high probability, which we take to mean with probability at least $1 - 1/n^c$ for all constants c. The running times of our algorithms is written in terms of n and m, which refer to an upper bound on the number of vertices and edges, respectively, over the entire sequence of updates. That is, for incremental algorithms, the running time is written in terms of the final values of n and m and for decremental algorithms the running time written is in terms of the initial values of n and m.

Each of our algorithms is written as a reduction to a black-box incremental or decremental approximation algorithm for *truncated SSSP*; that is, SSSP which provides a distance estimate for all nodes whose distance from the source is at most a given value k. For generality, our algorithms are written for directed graphs and use directed SSSP algorithms, however if the graph is undirected one can simply run an undirected SSSP algorithm instead. Let *out-*$\mathcal{A}_{inc}(u,k,\delta)$ (resp., *out-*$\mathcal{A}_{dec}(u,k,\delta)$) be an incremental (resp., decremental) algorithm that maintains for all v an estimate $d'(u,v)$ such that if $d(u,v) \le k$ then $(1-\delta)d(u,v) \le d'(u,v) \le d(u,v)$. Analogously, let *in-*$\mathcal{A}_{inc}(u,k,\delta)$ (resp., *in-*$\mathcal{A}_{dec}(u,k,\delta)$) be an incremental (decremental) algorithm that maintains an estimate $d'(v,u)$ for all v such that if $d(u,v) \le k$ then $(1-\delta)d(v,u) \le d'(v,u) \le d(v,u)$. We assume that after every update, these algorithms output all nodes whose distance estimate has changed. Let $T_{inc}(n,m,k,\delta)$ (resp., $T_{dec}(n,m,k,\delta)$) be the total time of out-$\mathcal{A}_{inc}(u,k,\delta)$ and in-$\mathcal{A}_{inc}(u,k,\delta)$ (resp., out-$\mathcal{A}_{inc}(u,k,\delta)$ and in-$\mathcal{A}_{inc}(u,k,\delta)$) (or the corresponding undirected algorithms).

The running times of our algorithms are written as the maximum of an expression over all values of the diameter D (or radius R) throughout the entire sequence of updates. Although the maximum value of D and R in a partially dynamic graph either occurs at the beginning or end of the update sequence, the maximum value of the running time expression could occur for any value of D or R.

Suppose we run in-$\mathcal{A}_{inc}$, in-$\mathcal{A}_{dec}$, out-$\mathcal{A}_{inc}$, or out-$\mathcal{A}_{dec}$ from a vertex v. Then, let $B_{out}(v,r)$ be the set of vertices u with $d'(v,u) \le r$.

For a subset $S \subseteq V$ of vertices and a vertex $v \in V$ we define $d(S,v) := \min_{s\in S} d(s,v)$. Similarly, $d(v,S) := \min_{s\in S} d(v,s)$. When the algorithms call for an approximation $d'(S,v)$ of $d(S,v)$, we add a dummy vertex x with an edge to every vertex in S and run out-$\mathcal{A}_{inc}$ (or out-$\mathcal{A}_{dec}$) from x; let $d'(S,v) = d'(x,v) - 1$. We define and maintain $d'(v,S)$ analogously by adding a dummy vertex with an edge from every vertex in S.

▷ Claim 9. For all $u \notin S$, $(1-2\delta)d(u,S) \le d'(u,S) \le d(u,S)$.

Proof. $(1-2\delta)d(u,S) = (1-2\delta)(d(u,x)-1) = d(u,x) - \delta d(u,x) - 1 + \delta(2 - d(u,x)) \le d(u,x) - \delta d(u,x) - 1 = (1-\delta)d(u,x) - 1 \le d'(u,x) - 1 = d'(u,S)$ and $d'(u,S) = d'(u,x) - 1 \le d(u,x) - 1 = d(u,S)$. ◁

For all of our algorithms for diameter and eccentricities, the bulk of the argument is to prove a lemma of the following form: if one is given values P' and ϵ such that P' is at most the true value P of the parameter of interest, then there is an algorithm that outputs an estimate $\hat{P}$ such that $\alpha(1-\epsilon)P' - \beta \le \hat{P} \le P$ for appropriate α and β. Lemma 10, whose proof we defer to the full version [5], states that a lemma of the above form suffices to prove our theorems. (In Lemma 10, the number k of parameters is 1 for the case of diameter and n for the case of eccentricities.) Lemma 10 also requires a fast constant-factor approximation for the parameter of interest in the static setting. Such algorithms exist for directed diameter and eccentricities in near-linear time.

Lemma 10 does not apply to radius since radius is a minimization problem, however an analogous lemma holds for radius; we defer it to the full version [5].

▶ **Lemma 10.** *Let $\pi_1, \pi_2, \ldots, \pi_k$ be a set of graph parameters (e.g. eccentricities). Suppose there is a static $\tilde{O}(T'(n,m))$ time algorithm that gives a constant-factor approximation for all π_i. Let $P_1, P_2, \ldots, P_k$ be the dynamically changing values of $\pi_1, \pi_2, \ldots, \pi_k$, respectively. Suppose there is an algorithm $\mathcal{P}$ that given a partially dynamic graph and values P' and $\epsilon > 0$, runs in total time $T(n,m,P',\epsilon)$ where T is a polynomial, and maintains a set of estimates $\hat{P}_1 \leq P_1, \ldots, \hat{P}_k \leq P_k$ such that for all i, if $P' \leq P_i$, then $\hat{P}_i \geq \alpha(1-\epsilon)P' - \beta$ for constants α and β.*

Let P_{min} and P_{max} be the minimum and maximum respectively over all P_i over the entire sequence of updates. Then there is an algorithm $\mathcal{P}'$ that given a partially dynamic graph and $\epsilon > 0$, runs in total time $\tilde{O}(T'(n,m) + \max_{P_{min} \leq P' \leq P_{max}} T(n,m,P',\epsilon)/\epsilon)$ and maintains estimates $\hat{P}_1, \ldots, \hat{P}_k$ such that for all i, $\alpha(1-\epsilon)P_i - \beta \leq \hat{P}_i \leq P_i$.

2.2 Lower bounds

Let $k \geq 2$. The k-Orthogonal Vectors Problem (k-OV) is as follows: given k sets $S_1, \ldots, S_k$, where each S_i contains n vectors in $\{0,1\}^d$, determine whether there exist $v_1 \in S_1, \ldots, v_k \in S_k$ so that their generalized inner product is 0, i.e. $\sum_{i=1}^{d} \prod_{j=1}^{k} v_j[i] = 0$. The k-OV Hypothesis is that k-OV requires $n^{k-o(1)}$ time in the word-RAM model of computation with $O(\log n)$ bit words, even for randomized algorithms.

The *unbalanced* version of k-OV has the sets S_i potentially have different sizes, $|S_i| = n_i$. The unbalanced k-OV Hypothesis is that unbalanced k-OV requires $(\prod_i n_i)^{1-o(1)}$ time. When each n_i is polynomial in n, the unbalanced k-OV Hypothesis is known to be equivalent to the k-OV Hypothesis.

R. Williams [25] (see also [24]) showed that if for some $\epsilon > 0$ there is an $n^{k-\epsilon}$poly (d) time algorithm for k-OV, then CNF-SAT on formulas with N variables and m clauses can be solved in $2^{N(1-\epsilon/k)}$poly (m) time. In particular, such an algorithm would contradict the Strong Exponential Time Hypothesis (SETH) of Impagliazzo, Paturi and Zane [16] which states that for every $\epsilon > 0$ there is a K such that K-SAT on N variables cannot be solved in $2^{(1-\epsilon)N}$poly N time (on a word-RAM with $O(\log N)$ bit words) even by randomized algorithms. Thus SETH implies the k-OV Hypothesis for all constants k. Each of our lower bounds conditional upon SETH is a reduction from either unbalanced 2 or 3-OV

The Hitting Set (HS) problem [2] is: given two sets U and V of n vectors each in $\{0,1\}^d$, is there $u \in U$ so that for all $v \in V$, $u \cdot v \neq 0$? The HS Hypothesis states that HS requires $n^{2-o(1)}$ time in the word-RAM model with $O(\log n)$ bit words, even for randomized algorithms.

We introduce the unbalanced version of HS, for three unbalanced sets. Unbalanced 3-HS is the problem, given $U, V, W \subseteq \{0,1\}^d$ with $|U| = n, |V| = n^a, |W| = n^b$ for constants $a, b > 0$, are there $u \in U, w \in W$ so that for all $v \in V$, $u \cdot v \cdot w \neq 0$? This is in similar spirit to unbalanced 3-OV. The unbalanced 3-HS Hypothesis is that unbalanced 3-HS requires $(|U| \cdot |V| \cdot |W|)^{1-o(1)} = n^{1+a+b-o(1)}$ time in the word-RAM model with $O(\log n)$ bit words, even for randomized algorithms. Due to its similarity to 3-OV and the lack of good algorithms, the 3-HS Hypothesis is believable. Refuting it would imply some very interesting improved algorithms for a balanced variant of Quantified Boolean Formulas with 2 quantifiers [18].

As mentioned in the introduction, the k-Cycle Hypothesis is that in the word-RAM model with $O(\log n)$ bit words, any possibly randomized algorithm needs $m^{2-f(k)-o(1)}$ time to find a k-cycle in an m-edge graph, for some continuous (over the reals) $f(k)$ that goes to 0 as k goes to infinity. The Hypothesis is completely consistent with the state of the art k-Cycle algorithms (e.g. [27, 4, 9]).

3 Nearly 3/2-approximation of Diameter

In this section we present a nearly 3/2-approximation for incremental/decremental diameter, given access to a black-box incremental/decremental approximate SSSP algorithm as specified in the preliminaries. We defer the remaining algorithms to the full version [5].

▶ **Theorem 11.** *There is a Las Vegas randomized algorithm for incremental (resp., decremental) diameter in unweighted, directed graphs against an oblivious (resp., adaptive) adversary that given $\epsilon > 0$, runs in $\tilde{O}(\max_{D_f \leq D' \leq D_0}\{T_{inc}(n, m, D', \epsilon)\frac{\sqrt{n/D'}}{\epsilon^2}\})$ (resp., $\tilde{O}(\max_{D_0 \leq D' \leq D_f}\{T_{dec}(n, m, D', \epsilon)\frac{\sqrt{n/D'}}{\epsilon^2}\})$) total time with high probability, and maintains an estimate $\hat{D}$ such that $\frac{2(1-\epsilon)}{3}D - \frac{2}{3} \leq \hat{D} \leq D$ where D is the diameter of the current graph.*

By Lemma 10, the following lemma implies Theorem 11.

▶ **Lemma 12.** *There is a Las Vegas algorithm for incremental (resp., decremental) diameter in unweighted, directed graphs against an oblivious (resp., adaptive) adversary that given $D', \epsilon > 0$, runs in $\tilde{O}\left(T_{inc}(n, m, D', \epsilon)\frac{\sqrt{n/D'}}{\epsilon}\right)$ (resp., $\tilde{O}\left(T_{dec}(n, m, D', \epsilon)\frac{\sqrt{n/D'}}{\epsilon}\right)$) total time with high probability, and maintains an estimate $\hat{D} \leq D$ such that if $D' \leq D$ then $\hat{D} \geq \frac{2(1-\epsilon)}{3}D' - \frac{2}{3}$ where D is the diameter of the current graph.*

Algorithm

Here we give the algorithm for Lemma 12. We defer the proof to the full version [5]. Let $\delta = 2\epsilon/11$. Throughout the incremental (resp., decremental) algorithm we will run in-$\mathcal{A}_{inc}$ (in-$\mathcal{A}_{dec}$) and out-$\mathcal{A}_{inc}$ (out-$\mathcal{A}_{dec}$) from certain sets of vertices. For ease of notation, we let in-$\mathcal{A}$ denote either in-$\mathcal{A}_{inc}$ or in-$\mathcal{A}_{dec}$, depending on the setting, and similarly we let out-$\mathcal{A}$ denote either out-$\mathcal{A}_{inc}$ or out-$\mathcal{A}_{dec}$.

Initialization. Let α be such that $D' = \Theta(n^{1-2\alpha})$. We randomly sample a set S of size $\Theta(n^\alpha \log^2 n)$ so that with high probability, for every vertex v, after every update, S hits $N_{out}(v, n^{1-\alpha})$. Throughout the entire execution of the algorithm, for all $s \in S$ we run in-$\mathcal{A}(s, D', \delta)$. Additionally, we maintain the approximate distance $d'(v, S)$ from every vertex v to S as described in the preliminaries. Let W be the dynamically changing set of vertices v that satisfy $d'(v, S) > D'/3$.

Phases. In the incremental setting on the other hand, distances can decrease so vertices can leave W. The incremental algorithm may have many phases, and at the beginning of each phase, we choose $w \in W$ uniformly at random. The beginning of a new phase is triggered when w leaves the set W.

Throughout the phase, we run out-$\mathcal{A}(w, D', \delta)$. Also, we will define a subset $S' \subseteq B_{out}(w, \frac{D'}{3})$ and for all $s' \in S'$, we run in-$\mathcal{A}(s', D', \delta)$. S' is initially empty and we independently add each vertex in $B_{out}(w, \frac{D'}{3})$ to S' with probability $\min\{1, \frac{\log^2 n}{\delta D'}\}$. In the incremental setting (but not the decremental setting), $B_{out}(w, \frac{D'}{3})$ can grow, and whenever a vertex u joins $B_{out}(w, \frac{D'}{3})$ we add u to S' with probability $\min\{1, \frac{\log^2 n}{\delta D'}\}$.

Reinitialization. We reinitialize the entire algorithm if at any point during the execution of the algorithm, any of the following occur: (1) $|B_{out}(w, \frac{D'}{3})| > n^{1-\alpha}$, (2) $|S'| > \frac{n^{1-\alpha}\log^4 n}{\delta D'}$, or (3) there is a vertex $v \in B_{out}(w, \frac{D'}{3})$ such that $d'(v, S') > \delta D'$. We will show in the analysis that with high probability we never reinitialize the algorithm.

Query. Following each update, the return value $\hat{D}$ is the maximum distance estimate found over all instantiations of out-$\mathcal{A}$ and in-$\mathcal{A}$. That is,
$\hat{D} = \max\{\max_{v\in V} d'(w, v), \max_{v\in V, s\in S\cup S'} d'(v, s)\}$.

To maintain this value, we maintain the following heaps. For every vertex v that we run out-$\mathcal{A}$ (resp., out-$\mathcal{A}$) from, we keep a max-heap $\mathcal{H}(v)$ that stores for each other vertex u the estimate $d'(v, u)$ (resp., $d'(u, v)$). Let $\hat{d}_{out}(v)$ be the value that $\mathcal{H}(v)$ outputs. Additionally we keep a max-heap $\mathcal{H}$ which stores each $\hat{d}_{out}(v)$.

4 $(2-\epsilon)$-approximation requires linear update time

In this section we give a linear lower bound per update for $(2-\epsilon)$-approximation for diameter, radius, and fixed-vertex eccentricity of undirected graphs. We defer the remaining proofs to the full version [5].

▶ **Theorem 13.** *Let t, ϵ, and ϵ' be positive constants. SETH implies that there exists no fully dynamic algorithm for $(2-\epsilon)$-approximate Diameter, Radius, or fixed-vertex Eccentricity on undirected, unweighted graphs with n vertices and $\tilde{O}(n)$ edges, which has preprocessing time $p(n) = O(n^t)$, amortized update time $u(n) = O(n^{1-\epsilon'})$, and amortized query time $q(n) = O(n^{1-\epsilon'})$. The same holds for the incremental/ decremental settings, for worst-case update and query times.*

Proof of Theorem 13.

Initialization

Let $a = \lceil\frac{2-\epsilon}{2\epsilon}\rceil + 1$ and $\delta = \frac{1-\epsilon'}{t}$. We begin with an instance of 2-OV with vector sets U and V of vectors, with $|U| = N^\delta$ and $|V| = N^{1-\delta}$. We create a graph G as follows. Add a node s and for each coordinate c, create two paths of length $2a$ beginning at s, and denote the endpoints of the paths by c_{left} and c_{right}.

Next, create two paths of length a for each vector $u \in U$. Denote the endpoints of one path by u^0_{left} and u^a_{left}, and the endpoints of the other by u^0_{right} and u^a_{right}. Finally, we encode each vector $u \in U$ in the graph by connecting u^a_{left} to c_{left} with a path of length a if $u[c] = 1$, and doing the same on the right side of G. If u has no coordinates equal to 1, then we may report that there is an orthogonal pair and halt; thus there will be no disconnected nodes in G.

Dynamic stages

We proceed in $N^{1-\delta}$ stages, one for each element $v \in V$. For the current v, for each coordinate c where $v[c] = 1$, we add edges (s, c_{left}) and (s, c_{right}). We then query the diameter or radius of G or the eccentricity of s. We will show that the eccentricity of s is always equal to the radius, and that if the diameter is least $8a$ or the radius is at least $4a$, then there is an orthogonal pair u, v; otherwise, the diameter is at most $4a + 2$ or the radius is at most $2a + 1$. We have set a so that a $(2-\epsilon)$-approximation algorithm for diameter, radius or eccentricity of s distinguishes between these two cases and thus detects an orthogonal pair. If the query does not detect an orthogonal pair, we undo the edge additions for the stage and continue to the next v. We can modify the stage to be decremental by beginning with edges from s to all nodes c_{left} and c_{right}, and removing the excess edges each stage.

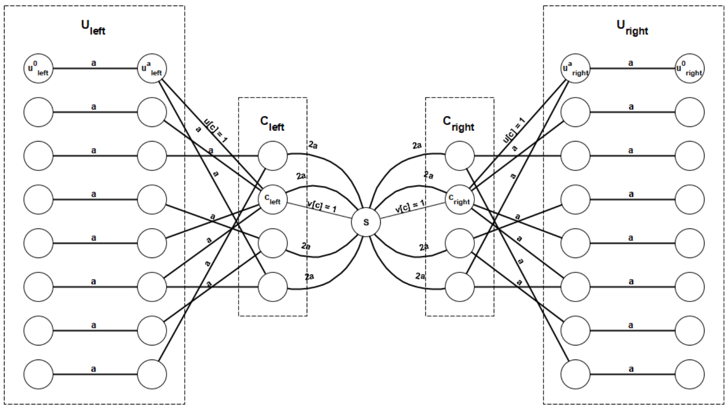

Figure 1 Sketch of Theorem 13 Construction. Bold edges represent paths, whose labels denote their length.

Correctness

We claim that the node s is always the center of G, so the eccentricity of s is always the radius of G. Let x_{right} be the node farthest from s on the right side of G. Since the graph is symmetrical, the counterpart x_{left} of x_{right} is such that $d(s, x_{right}) = d(s, x_{left})$. Any node y to the left of s must pass through s to reach x_{right}, so y has a higher eccentricity than s because it is farther from the node farthest from s. Symmetrically, any node y to the right of s must pass through s to reach x_{left}, so y has a higher eccentricity than s because it is farther from the node farthest from s.

If for the current stage, for all u, $u \cdot v \neq 0$, then for each u there must be some coordinate c such that $u[c] = v[c] = 1$. Then there is a path of length 1 from s to c_{left}, and a path of length a from c_{left} to u^a_{left}, for all u. The same is true on the right side. Then since all nodes except s are of distance at most a from a node u'^a_{left} or u'^a_{right} for some $u' \in U$, all nodes are accessible in at most $2a+1$ steps from s. This means that the radius and eccentricity of s is $2a+1$, and the diameter is at most $4a+2$.

If for the current stage there is some u such that $u \cdot v = 0$, then there is no direct path from s to u^0 on either side via a vector coordinate c and u^a. A path via a different u'_a would be of length at least $4a+1$, because returning to a c' where $u[c'] = 1$ would cost an additional $2a$ from the direct path. The shortest path would thus be along the length-$2a$ path from s to c', giving $d(s, u^0) = 4a$. The radius and eccentricity of s must be at least $4a$ and diameter must be at least $8a$, because $d(u^0_{left}, u^0_{right}) = d(s, u^0_{left}) + d(s, u^0_{right}) = 4a + 4a = 8a$.

Running time

We assume for the sake of contradiction that the algorithm of Theorem 13 exists. Let $n = N^\delta$ be the size of G. We have that $u(n) = q(n) = O((N^\delta)^{1-\epsilon'})$. After initialization and $|V| = N^{1-\delta}$ stages, the total update and query time is then at most $\tilde{O}(N^{1-\delta\epsilon'})$. The preprocessing time $p(n)$ for the algorithm on G is $O((N^\delta)^t) = O(N^{1-\epsilon'})$. Thus the total time of the algorithm is $\tilde{O}(N^{1-\epsilon'} + N^{1-\delta\epsilon'})$. This contradicts SETH, because SETH implies that no algorithm exists for 2-OV in $O((|U| \cdot |V|)^{1-\epsilon''}) = O(N^{1-\epsilon''})$ time for any $\epsilon'' > 0$. ◀

References

1 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.

2 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 377–391, 2016.

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

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

5 Bertie Ancona, Monika Henzinger, Liam Roditty, Virginia Vassilevska Williams, and Nicole Wein. Algorithms and Hardness for Diameter in Dynamic Graphs. *arXiv preprint*, 2018. arXiv:1811.12527.

6 Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards Tight Approximation Bounds for Graph Diameter and Eccentricities. In *Proceedings of STOC'18*, page to appear, 2018.

7 Massimo Cairo, Roberto Grossi, and Romeo Rizzi. New Bounds for Approximating Extremal Distances in Undirected Graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 363–376, 2016.

8 Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert Endre Tarjan, and Virginia Vassilevska Williams. Better Approximation Algorithms for the Graph Diameter. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1041–1052, 2014.

9 Mina Dalirrooyfard, Thuy Duong Vuong, and Virginia Vassilevska Williams. Graph pattern detection: Hardness for all induced patterns and faster non-induced cycles. In *STOC*, page to appear, 2019.

10 Camil Demetrescu and Giuseppe F. Italiano. A New Approach to Dynamic All Pairs Shortest Paths. *Journal of the ACM*, 51(6):968–992, 2004. Announced at STOC'03. doi:10.1145/1039488.1039492.

11 Shimon Even and Yossi Shiloach. An On-Line Edge-Deletion Problem. *Journal of the ACM*, 28(1):1–4, 1981. doi:10.1145/322234.322235.

12 Monika Henzinger and Valerie King. Fully Dynamic Biconnectivity and Transitive Closure. In *Symposium on Foundations of Computer Science (FOCS)*, pages 664–672, 1995. doi:10.1109/SFCS.1995.492668.

13 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Decremental Single-Source Shortest Paths on Undirected Graphs in Near-Linear Total Update Time. In *Symposium on Foundations of Computer Science (FOCS)*, pages 146–155, 2014. doi:10.1109/FOCS.2014.24.

14 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture. In *Symposium on Theory of Computing (STOC)*, pages 21–30, 2015. doi:10.1145/2746539.2746609.

15 Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. Conditional Hardness for Sensitivity Problems. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:31, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2017.26.

16 R. Impagliazzo, R. Paturi, and F. Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

17 François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014.

18 Moshe Lewenstein, Seth Pettie, and Virginia Vassilevska Williams. Open Problems from Dagstuhl Seminar 16451: Structure and Hardness in P, 2016.

19 S. Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theor. Comput. Sci.*, 312(1):47–74, 2004.

20 Seth Pettie and Vijaya Ramachandran. A Shortest Path Algorithm for Real-Weighted Undirected Graphs. *SIAM J. Comput.*, 34(6):1398–1431, 2005.

21 Liam Roditty and Virginia 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, 2013.

22 R. Seidel. On the All-Pairs-Shortest-Path Problem in Unweighted Undirected Graphs. *J. Comput. Syst. Sci.*, 51(3):400–403, 1995.

23 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 887–898. ACM, 2012.

24 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians*, page to appear, 2018.

25 R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2–3):357–365, 2005.

26 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 664–673, 2014.

27 R. Yuster and U. Zwick. Detecting short directed cycles using rectangular matrix multiplication and dynamic programming. In *Proc. SODA*, pages 247–253, 2004.

28 Uri Zwick. All Pairs Shortest Paths using Bridging Sets and Rectangular Matrix Multiplication. *Journal of the ACM*, 49(3):289–317, 2002. Announced at FOCS'98. `doi:10.1145/567112.567114`.