Approximation Algorithms for Min-Distance Problems

Mina Dalirrooyfard
MIT, Cambridge, MA, USA
minad@mit.edu

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

Nikhil Vyas
MIT, Cambridge, MA, USA
nvyas@mit.edu

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

Yinzhan Xu
MIT, Cambridge, MA, USA
xyzhan@mit.edu

Yuancheng Yu
MIT, Cambridge, MA, USA
ycyu@mit.edu

Abstract

We study fundamental graph parameters such as the Diameter and Radius in directed graphs, when distances are measured using a somewhat unorthodox but natural measure: the distance between u and v is the *minimum* of the shortest path distances from u to v and from v to u. The center node in a graph under this measure can for instance represent the optimal location for a hospital to ensure the fastest medical care for everyone, as one can either go to the hospital, or a doctor can be sent to help.

By computing All-Pairs Shortest Paths, all pairwise distances and thus the parameters we study can be computed exactly in $\tilde{O}(mn)$ time for directed graphs on n vertices, m edges and nonnegative edge weights. Furthermore, this time bound is tight under the Strong Exponential Time Hypothesis [Roditty-Vassilevska W. STOC 2013] so it is natural to study how well these parameters can be *approximated* in $O(mn^{1-\varepsilon})$ time for constant $\varepsilon > 0$. Abboud, Vassilevska Williams, and Wang [SODA 2016] gave a polynomial factor approximation for Diameter and Radius, as well as a constant factor approximation for both problems in the special case where the graph is a DAG. We greatly improve upon these bounds by providing the first constant factor approximations for Diameter, Radius and the related Eccentricities problem in general graphs. Additionally, we provide a hierarchy of algorithms for Diameter that gives a time/accuracy trade-off.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases fine-grained complexity, graph algorithms, diameter, radius, eccentricities

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.46

Category Track A: Algorithms, Complexity and Games

Related Version https://arxiv.org/abs/1904.11606

Acknowledgements The authors would like to thank the members of the MIT course 6.S078 open problem sessions, especially Thuy-Duong Vuong, Robin Hui, and Ali Vakilian. These sessions were organized by Erik Demaine, Ryan Williams, and Virginia Vassilevska Williams.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 46; pp. 46:1–46:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The diameter, radius and eccentricities of a graph are fundamental parameters that have been extensively studied [13, 20, 12, 18, 3, 14, 11, 17, 5, 6, 26, 27, 9, 19, 24, 23, 10, 1, 7] (and many others). The eccentricity of a vertex v is the largest distance between v and any other vertex. The diameter is the maximum eccentricity of a vertex in the graph, thus measuring how far apart two nodes can be, and the radius is the minimum eccentricity, measuring the maximum distance to the most central node.

The distance between two vertices in an undirected graph is just the shortest path distance $d(\cdot,\cdot)$ between them. For directed graphs, however, this notion of distance d is no longer necessarily symmetric, and rather than being a distance *between* two nodes, it measures the distance in a given direction. Several related notions of pairwise distance that are symmetric have been studied. These include the roundtrip distance [15] which for two vertices u and v is just $d(u,v)+d(v,u)$, the max-distance [2] which is $\max\{d(u,v),d(v,u)\}$, and the min-distance [2] which is $\min\{d(u,v),d(v,u)\}$.

Each of these notions of distance has a particular application. For instance, one would have to pay the roundtrip distance when going to the store and back. On the other hand, if one needs medical assistance, one could either go to the hospital, or have a physician come to the home – the time to receive care is then measured by the min-distance. Another example of min-distance is in symmetric-key encryption: any pair of parties can create a shared private key by using only one-way communication.

For each notion of distance, the diameter, radius and eccentricity parameters are well-defined. Given the shortest path distances $d(\cdot,\cdot)$ for all vertices, the parameters for each distance measure can be computed in $O(n^2)$ time in n vertex graphs. The fastest known algorithms for All-Pairs Shortest Paths (APSP) [25, 21, 22] give the fastest known algorithms to compute these parameters exactly, running in $n^3/\exp(\sqrt{\log n})$ time and $O(mn+n^2\log\log n)$, respectively on m-edge, n-vertex graphs. Furthermore, under the Strong Exponential Hypothesis, there is no $O(m^{2-\varepsilon})$ time algorithm for Diameter in unweighted graphs (and thus also for any of these notions of Diameter and Eccentricities in directed graphs) [23]. For Radius, the same lower bound holds but under the "Hitting Set" conjecture [2].

As exact computation is expensive, it makes sense to resort to approximation algorithms. For the shortest path distance versions of Diameter, Eccentricities and Radius, there are several fast algorithms that achieve various small constant approximation ratios [23, 10, 8, 4]. For instance, for Diameter, a folklore linear time algorithm can achieve a 2-approximation, and an $\tilde{O}(m^{3/2})$ time[1] algorithm can achieve a 3/2-approximation [23, 10].

Many of these algorithms [23, 10, 4] work for any distance measure that satisfies the triangle inequality. Thus they work for the shortest paths distance, max-distance and roundtrip distance. The min-distance however does not satisfy the triangle inequality: e.g. you might have edges (x,y) and (z,y), and thus the min-distance between x and y and between y and z are both 1, yet there may be no directed path between x and z in any direction, so that the min-distance between them may be ∞.

This issue makes it much more difficult to design fast approximation algorithms for Min-Diameter, Min-Radius and Min-Eccentricities (the parameters of interest under the min-distance). The only known nontrivial algorithms are by Abboud et al. [2]. For Min-Diameter [2] gives a near-linear time 2-approximation algorithm if the input is a directed acyclic graph. For general graphs, the only nontrivial fast approximation algorithm is an

[1] We use $\tilde{O}$ notation to hide polylogarithmic factors.

$\tilde{O}(mn^{1-\varepsilon})$ time n^{ε}-approximation algorithm for any constant $\varepsilon > 0$. (No constant factor approximation algorithm is known that runs significantly faster than just computing APSP.) For Min-Radius, [2] gives an $\tilde{O}(m\sqrt{n})$ time 3-approximation algorithm for directed acyclic graphs. For general graphs, they only achieve a very weak n-approximation in near-linear time that checks if the Min-Radius is finite. There are no known approximation algorithms for Min-Eccentricities faster than just computing APSP.

1.1 Our Results

The main goal of our paper is to obtain new fast, $O(mn^{1-\varepsilon})$ time for some constant $\varepsilon > 0$, algorithms for Min-Diameter, Min-Radius and Min-Eccentricities (thus beating the $\tilde{O}(mn)$ time of exact computation). We achieve this by developing powerful new techniques that can handle the complications that arise due to the fact that the min-distance does not satisfy the triangle inequality.

Our results are as follows. For Min-Diameter we achieve a hierarchy of algorithms trading off running time with approximation accuracy.

▶ **Theorem 1.** *For any integer $0 < \ell \leq O(\log n)$, there is an $\tilde{O}(mn^{1/(\ell+1)})$ time randomized algorithm that, given a directed weighted graph G with edge weights non-negative and polynomial in n, can output an estimate $\tilde{D}$ such that $D/(4\ell - 1) \leq \tilde{D} \leq D$ with high probability, where D is the min-diameter of G.*

When we set $\ell = 1$, we obtain an $\tilde{O}(m\sqrt{n})$ time 3-approximation algorithm, and when we set $\ell = \lceil \log n \rceil$, we get an $\tilde{O}(m)$ time $O(\log n)$-approximation.

Our tradeoff achieves the first constant factor approximation algorithms for Min-Diameter in general graphs that run in $O(mn^{1-\varepsilon})$ time for constant $\varepsilon > 0$. Such a result was only known for directed acyclic graphs, whereas for general graphs the only known efficient algorithm could achieve an n^{ε}-approximation.

For Min-Radius, we also achieve the first constant factor approximation algorithm for general graphs running in $O(mn^{1-\varepsilon})$ time for some constant $\varepsilon > 0$. Such a result was only known for directed acyclic graphs, whereas for general graphs the only known efficient algorithm could only check if the Min-Radius is finite.

▶ **Theorem 2.** *For any constant δ with $1 > \delta > 0$, there is an $\tilde{O}(m\sqrt{n}/\delta)$ time randomized algorithm, that given a directed weighted graph G with edge weights positive and polynomial in n, can output an estimate R' such that $R \leq R' \leq (3+\delta)R$ with high probability, where R is the min-radius of G.*

Finally, we obtain the first $O(mn^{1-\varepsilon})$ time (for constant $\varepsilon > 0$) constant factor approximation algorithms for the Min-Eccentricities of all vertices in a graph. For unweighted graphs we are able to obtain a close to 3 approximation in $\tilde{O}(m\sqrt{n})$ time. For weighted graphs, our approximation factor grows to 5, while the running time is the same. Previously, the only algorithm to approximate the Min-Eccentricities computed them exactly via an APSP computation.

▶ **Theorem 3.** *For any constant δ with $1 > \delta > 0$, there is an $\tilde{O}(m\sqrt{n}/\delta)$ time randomized algorithm, that given a directed weighted graph $G = (V, E)$ with weights positive and polynomial in n, can output an estimate $\varepsilon'(s)$ for every vertex $s \in V$ such that $\varepsilon(s) \leq \varepsilon'(s) \leq (5+\delta)\varepsilon(s)$ with high probability, where $\varepsilon(s)$ is the min-eccentricity of vertex s in G.*

▶ **Theorem 4.** *For any constant δ with $1 > \delta > 0$, there is an $\tilde{O}(m\sqrt{n}/\delta^2)$ time randomized algorithm, that given a directed unweighted graph $G = (V, E)$, can output an estimate $\varepsilon'(s)$ for every vertex $s \in V$ such that $\varepsilon(s) \leq \varepsilon'(s) \leq (3+\delta)\varepsilon(s)$ with high probability, where $\varepsilon(s)$ is the min-eccentricity of the vertex s in G.*

1.2 Our Techniques

To obtain our results, we develop powerful new techniques which we outline below.

Partial search graphs. The idea of partial search graphs is used in the algorithms of [2] for Min-Radius and Min-Diameter on DAGs. These algorithms use the following high-level framework: perform Dijkstra's algorithm from some vertices and then perform a *partial* Dijkstra's algorithm from *every* vertex. The partial search from a vertex v is with respect to a carefully defined partial search graph $G_v \subset G$. The crux of the analysis for the algorithms on DAGs is to argue that if the executions of Dijkstra's algorithm on the full graph did not find a good estimate for the desired quantity (either min-diameter or min-radius), then the partial search from some vertex v returns a good estimate of the min-eccentricity of v, which in turn is a good estimate for the desired quantity. In DAGs it is natural to define the partial search graphs G_v by considering a topological ordering of the vertices and letting each G_v be some interval containing v (though defining the exact intervals requires some work). For general graphs it is completely unclear how to even define such intervals since there is no natural notion of an ordering of the vertices, and thus figuring out what the G_v's should be is nontrivial. Our approach to overcoming this hurdle is to carefully define a DAG-like structure in general graphs. Such a structure may be of independent interest.

Defining a DAG-like structure in general graphs. It would be ideal to directly reduce the problem on general graphs to the problem on DAGs, however it is very unclear how to do this. Instead, we recognize that it suffices to define a *DAG-like* structure in general graphs. As a first step, we use the following idea. Suppose we have performed Dijkstra's algorithm from a vertex v. We let $S_v = \{u : d(u,v) < d(v,u)\}$ and we let $T_v = \{u : d(u,v) > d(v,u)\}$[2]. Then, we partially order the vertices so that the vertices in S_v appear before v and those in T_v appear after v. We note that this partial ordering is "DAG-like" because it is consistent with the topological ordering of a DAG; that is, if we apply this partition into S_v and T_v to a DAG then there trivially exists a topological ordering such that every vertex in S_v appears before v and every vertex in T_v appears after v. After partitioning into S_v and T_v, we recursively partition each set to create a more precise partial ordering. Importantly, we show that by recursively sampling vertices randomly, we can guarantee that our partitioning is approximately balanced which is crucial for the runtime analysis. The obtained partial ordering is the starting point for all of our algorithms.

Min-Diameter: graph augmentation. The Min-Diameter algorithm on DAGs from [2] relies heavily on the following key property of DAGs. Consider a topological ordering and the graphs induced by the first and second halves of the ordering; which are defined with respect to the middle vertex in the ordering. For all pairs of vertices in the same half of the ordering, their min-distance in the graph induced by this half is the same as their min-distance in the full graph. As previously mentioned, if we sample a vertex v, we can make sure that S_v and T_v are approximately balanced, so that we can think of S_v and T_v as corresponding to the first and second half of a DAG topological ordering, respectively. However it is unclear how to obtain a property of S_v and T_v analogous to the above key property of DAGs. In particular, the min-distance between a pair of vertices in the graph induced by S_v could be wildly different from their min-distance in the full graph, since paths whose endpoints are

[2] u's with $d(u,v) = d(v,u)$ are added to either S_v or T_v as specified in the formal definition later

in S_v can contain vertices outside of S_v. To overcome this hurdle, we *augment* the graph induced by S_v and the graph induced by T_v by carefully adding edges so that distances within these augmented graphs approximate distances in the original graph.

Min-Radius: refined DAG-like structure. Our Min-Radius algorithm is much more delicate than our Min-Diameter algorithm due to the fact that for Min-Radius we care about small distances instead of large distances. In particular, the graph augmentation idea from our Min-Diameter algorithm does not help for Min-Radius because although the augmentations do not distort large distances much, they heavily distort small distances. Furthermore, the previously mentioned DAG-like structure for general graphs does not suffice for Min-Radius. However we use it as a starting point to define a more refined DAG-like partial ordering. Most of our algorithm is concerned with precisely arranging vertices in this partial ordering. Specifically, we structure the partial ordering to satisfy *roughly* the following property: for every pair of vertices u, v such that u appears before v in the partial ordering, $d(v,u)$ is large while $d(u,v)$ is small.

1.3 Notation

Given a graph $G=(V,E)$, $n=|V|$ and $m=|E|$. Graphs are directed and have non-negative weights polynomial in n unless otherwise specified. For any pair of vertices u and v, the *distance from u to v* $d(u,v)$ is the length of the shortest directed path from u to v. When the context is not clear, we write $d_G(u,v)$ to specify the graph G. The *min-distance* between a pair of vertices u and v is $d_{min}(u,v)=\min\{d(u,v),d(v,u)\}$. The *min-diameter* of a graph is $\max_{u,v\in V} d_{min}(u,v)$. The *min-radius* of a graph is $\min_{v\in V}\max_{u\in V} d_{min}(u,v)$. For any vertex v, the *min-eccentricity* of v is $\varepsilon(v)=\max_{u\in V} d_{min}(u,v)$. When the context is not clear, we say $\varepsilon_G(v)$ to specify the graph G. Note that we do not use the *min* subscript to denote the min-eccentricity of a vertex. For an algorithm with input size n we use *with high probability* to denote the probability $>1-1/n^c$ for all constants c. We say some quantity is *poly*(n) to mean it is $O(n^c)$ for some fixed constant c. We use $\tilde{O}$ notation to hide polylogarithmic factors.

1.4 Organization

In Section 2 we give an overview of all of our algorithms, in Section 3 we describe a graph partitioning procedure that begins all of our algorithms, in Section 4 we describe our Min-Diameter algorithms. We defer the time/accuracy tradeoff algorithm for Min-Diameter, the Min-Radius algorithm and the Min-Eccentricities algorithm to the full version [16].

2 Overview of Algorithms

We use the algorithms from [2] for Min-Diameter and Min-Radius on DAGs as inspiration. For each problem, we first outline the DAG algorithm and then provide intuition for how to apply these ideas to general graphs.

2.1 Min-Diameter

Algorithm for DAGs

We begin by outlining the $\tilde{O}(n+m)$ time 2-approximation algorithm for Min-Diameter on DAGs from [2]. Consider a topological ordering of the vertices and perform Dijkstra's algorithm from the middle vertex v. Then recurse on the graphs induced by the vertices in

the first half (before v) and in the second half (after v). A key observation in the analysis is that if the true endpoints s^* and t^* of the min-diameter fall on opposite sides of v in the ordering, then the min-eccentricity $\varepsilon(v)$ of v is a 2-approximation for the min-diameter D. This is because if $\varepsilon(v) < D/2$ and s^* and t^* fall on opposite sides of v in the ordering, then $d(s^*, v) < D/2$ and $d(v, t^*) < D/2$ so $d(s^*, t^*) < D$, a contradiction. So, suppose (without loss of generality) that s^* and t^* both fall before v in the ordering. Since the graph is a DAG, every path between s^* and t^* only uses vertices before v in the ordering. Thus, the min-distance between s^* and t^* in the graph induced by the first half of the graph is still D.

Algorithm for general graphs

We now outline a precursor to our Min-Diameter algorithm for general graphs that mimics the algorithm for DAGs. This $\tilde{O}(n+m)$ time algorithm does not achieve a constant approximation factor, however it provides intuition for our constant-factor approximation algorithms. We begin by performing Dijkstra's algorithm from a vertex v and constructing S_v and T_v as defined in the previous section. Analogously to the DAG algorithm if the true min-diameter endpoints s^* and t^* fall into different sets S_v, T_v then the min-eccentricity $\varepsilon(v)$ is a 2-approximation. This is because if $\varepsilon(v) < D/2$, $s^* \in S_v$, and $t^* \in T_v$ then $d(s^*, v) < D/2$ and $d(v, t^*) < D/2$ so $d(s^*, t^*) < D$, a contradiction. However, unlike the DAG algorithm, we cannot simply recurse independently on the graphs induced by S_v and T_v since the shortest path between a pair of vertices in S_v may not be completely contained in S_v (and analogously for T_v).

To overcome this hurdle, before recursing we first augment the graphs induced by S_v and T_v by carefully adding edges so that distances within these augmented graphs approximate distances in the original graph. Specifically, for every vertex $u \in S_v$, we add the directed edge (u, v) with weight 0 and the directed edge (v, u) with weight $\max\{0, d(v, u) - \varepsilon(v)\}$. This choice of edges allows us to argue that the distances within the augmented graphs are approximations of the distances in G up to an additive error of $2\varepsilon(v)$. Then, by returning the maximum of $\varepsilon(v)$ and the min-diameter estimates from recursing on the augmented graphs, we get an approximation guarantee, which turns out to be a logarithmic factor. Intuitively, the approximation factor is not constant because the recursion causes the distance distortion to compound at each level of recursion.

To reduce the approximation factor to a constant, we would like to decrease the number of recursion levels. To achieve this, we initially partition the graph into more than just two parts S_v and T_v, by sampling more vertices. For our $\tilde{O}(m\sqrt{n})$ time 3-approximation, we perform a full Dijkstra's algorithm from $\tilde{O}(\sqrt{n})$ vertices to define an ordered partition of the vertices into $\tilde{O}(\sqrt{n})$ parts of $\tilde{O}(\sqrt{n})$ vertices each. Then we apply the above idea of adding weighted edges within each part, however we must refine the definition of the graph augmentation to take into account *all* of the $\tilde{O}(\sqrt{n})$ vertices we initially perform Dijkstra's algorithm from, instead of just v. Finally we use brute force (without recursion) on each part in the partition by running an exact all-pairs shortest paths algorithm.

To achieve our time-accuracy trade-off algorithm, we carefully combine ideas from the logarithmic factor approximation and the 3-approximation algorithms. Specifically, we initially perform Dijkstra's algorithm from fewer than $\sqrt{n}$ vertices to define an ordered partition with larger parts than in the 3-approximation. Then we augment the graph induced by each part and carry out a constant number of recursion levels to further partition the graph before applying brute-force.

2.2 Min-Radius

Algorithm for DAGs

We begin by outlining the $\tilde{O}(m\sqrt{n})$ time 3-approximation algorithm for Min-Radius on DAGs from [2], which is very different from and more involved than the Min-Diameter algorithm on DAGs. We begin by considering a topological ordering of the vertices and performing Dijkstra's algorithm from a set W of $\tilde{O}(\sqrt{n})$ evenly spaced vertices including the first and last vertex. If a vertex $v \in W$ has min-eccentricity at most twice the true min-radius R then we have obtained a 2-approximation. (We do not know R in advance but we repeatedly run the algorithm with different values of R to perform a binary search on R.)

Otherwise, we will define intervals in the ordering such that the min-center c cannot be contained in any of these intervals. A key observation is that if there is a pair of vertices (u, v) such that u appears before v in the topological ordering and $d(u, v) > 2R$, then the min-center c cannot fall between u and v in the topological ordering. This is because if it did, then $d(u, c) \leq R$ and $d(c, v) \leq R$, so $d(u, v) \leq 2R$, a contradiction. We define the intervals that cannot contain c as follows: for all $v \in W$ we let a_v be the first vertex in the ordering such that $d(a_v, v) > 2R$ (if it exists, otherwise $a_v = v$) and define b_v to be the last vertex in the ordering such that $d(v, b_v) > 2R$ (if it exists, otherwise $b_v = v$). Then, the key observation implies that c cannot fall in the interval $[a_v, b_v]$ in the ordering. Now, we have a set of possibly overlapping intervals that cannot contain c. We take the union of these intervals to get a set of disjoint intervals that cannot contain c.

Every vertex u that does not appear in such an interval, falls between two consecutive intervals I_u and I'_u. We define the partial search graph of u to be the graph induced by the set of vertices in I_u or I'_u or between I_u and I'_u. After performing the partial searches, the algorithm returns 3 times the minimum min-radius of all partial search graph. Next we give the idea of the analysis, which demystifies the factor of 3 in the returned value.

We claim that if the min-eccentricity of a vertex with respect to its partial search graph is at most R, then its min-eccentricity with respect to the full graph is at most $3R$, and the min-eccentricity of the true min-center with respect to its partial search graph is at most R (because for any path in a DAG whose starting and ending points are in a certain interval, every vertex in the path is in that interval). Thus, assuming the claim, $3R$ is a 3-approximation for the min-radius. We now outline the proof of the claim. Let u be the min-center with the minimum min-radius R of all partial search graphs. Let $v \in W$ such that a_v is the first vertex (in the topological order) of I_u, then $v \in I_u$ and $d(v, u) \leq R$. Furthermore, by the definition of a_v, all vertices that appear before the beginning of the interval I_u have distance at most $2R$ to v, and thus distance at most $3R$ to u. A symmetric argument holds for vertices that appear after the end of the interval I'_u. Hence the min-eccentricity of u with respect to the full graph is at most $3R$.

This algorithm runs in time $O(m\sqrt{n})$ because the vertices of W are evenly spaced so there are no more than $\sqrt{n}$ vertices between each pair of consecutive intervals. This implies that in the partial searches, each edge is only scanned $O(\sqrt{n})$ times. (Furthermore, repeatedly running the algorithm to binary search for R adds a logarithmic factor to the runtime.)

Algorithm for general graphs

We now give a high-level outline of our $\tilde{O}(m\sqrt{n})$ time 3-approximation algorithm for Min-Radius. This algorithm is much more delicate than our Min-Diameter algorithm, hence more of the details are deferred to the full description. We begin by running Dijkstra's algorithm

from a set W of $\tilde{O}(\sqrt{n})$ randomly sampled vertices to recursively partition the vertices into S_v and T_v as outlined in Section 1.2. This defines an initial DAG-like structure, however our analysis requires constructing a much more refined DAG-like structure.

Perhaps counter-intuitively, it makes sense to place vertices that are *far* from each other in the graph *close* to each other in the DAG-like structure. The reason for this is illuminated by the Min-Radius algorithm on DAGs, in which we find pairs of vertices u, v that are far from each other and apply the key observation that the min-center cannot be between u and v in the topological ordering. Intuitively, it is as if we collapse the interval between u and v in the DAG since we do not have to search within this interval for the min-center. An analogous key observation is true for general graphs: if there is a pair of vertices (u, v) with $d_{min}(u, v) > 2R$, then either $c \in S_u \cap S_v$ or $c \in T_u \cap T_v$. This is because if $c \in T_u \cap S_v$, then $d(u, c) \leq R$ and $d(c, v) \leq R$ so $d(u, v) \leq 2R$, a contradiction; the last case $c \in S_u \cap T_v$ is symmetric. In our algorithm for general graphs, we ensure that far vertices are near each other in the DAG-like structure by doing the following: we let the *far graph* G_{far} be an undirected graph on V with an edge between $u \in W$ and $v \in V$ if $d_{min}(u, v) > 2R$. All vertices in W that are in the same connected component in G_{far} will be grouped in the DAG-like structure. We let F_i be the set of vertices in W that are in the i^{th} connected component of G_{far}.

To construct the DAG-like structure, we show that precisely chosen groups of F_is can be merged to create *supercomponents*, which constitute a DAG-like structure in the following sense: there is an ordering of supercomponents such that for every pair of vertices $u, v \in W$ where the supercomponent containing u appears before that containing v, $d(u, v)$ is small and $d(v, u)$ is large. Specifically, we define the *close graph* H whose vertex set is the set of F_is. We add a directed edge between a pair of vertices in H if there exists a short path (length $\leq 5R$) between the corresponding F_is. Then we merge all F_is that appear in the same strongly connected component of H into a supercomponent. This contraction of strongly connected components of H results in a DAG, which defines the ordering of the supercomponents.

Now that we have arranged the vertices in W into a DAG-like structure, we would like to fit every vertex in the graph into this structure. Based on the precise way that we have defined the supercomponents, we can use an intricate argument to show *roughly* the following property: for every vertex v there exists an i such that for every vertex $u \in W$ in the first i supercomponents, $d(u, v)$ is small and for every vertex $u \in W$ in the remaining supercomponents, $d(v, u)$ is small.

After fitting every vertex into the refined DAG-like ordering, we can define each partial search graph to be an interval in the ordering that is large enough to contain several supercomponents. In the algorithm for DAGs, there were two important properties of the partial search graphs: (1) the min-eccentricity of the true min-center with respect to its partial search graph is at most R, and (2) if the min-eccentricity of a vertex with respect to its partial search graph is at most R then its min-eccentricity with respect to the full graph is at most $3R$. We show that due to the precise structure of the supercomponents, refinements of properties (1) and (2) are also true for general graphs.

Intuitively, property (1) is roughly true because for every pair of vertices $u, v \in W$ such that u's supercomponent appears before v's in the ordering, $d(v, u) > 5R$, since otherwise this pair of supercomponents would be in the same strongly connected component of H and would have been merged into a single supercomponent. This implies that paths of length at most R to or from the min-center cannot stray beyond its partial search graph. Intuitively, property (2) is roughly true because for every pair of vertices $u, v \in W$ such that u's supercomponents appears before v's in the ordering, $d(u, v) \leq 2R$ because otherwise, u

and v would be in the same component of G_{far} and thus be in the same supercomponent. Thus, like the argument for DAGs, for all u, all vertices that appear before u's partial search graph G_u have distance at most $2R$ to each supercomponent in G_u, and thus distance at most $3R$ to u. A symmetric argument holds for vertices after u in the ordering.

2.3 Min-Eccentricities

Our Min-Eccentricities algorithm is a modification of our Min-Radius algorithm. In our Min-Radius algorithm, we identify a vertex whose min-eccentricity is at most about $3R$, where R is the true min-radius. In our Min-Eccentricities algorithm, we show that with some extra bookkeeping, the algorithm can identify *all* vertices with min-eccentricity at most about 5ρ for any ρ. We run the algorithm repeatedly, increasing ρ by a factor of $(1+\delta)$ at each execution until we have estimated the min-eccentricity of every vertex.

The major modification of the Min-Radius algorithm here is that if one of the vertices that we run Dijkstra from has min-eccentricity at most 3ρ, we cannot stop running the algorithm, as we can in the Min-Radius algorithm. Instead, we use this vertex as a tool to find vertices with min-eccentricity at most 5ρ.

3 Preliminary Graph Partitioning

In this section we describe a graph partitioning procedure we use as a first step in our Min-Diameter, Min-Radius, and Min-Eccentricities algorithms. The goal of this partitioning is to define a DAG-like structure in general directed graphs.

▶ **Definition 5.** *Assign each vertex a unique ID from* $[n]$*. For each vertex* v*, let* $S_v = \{u \in V : d(u,v) < d(v,u) \vee [d(u,v) = d(v,u) \wedge ID(u) < ID(v)]\}$*. Let* $T_v = V \setminus (S_v \cup \{v\})$*.*

The runtime of our algorithms relies on whether the partition into S_v and T_v is *balanced*. Using the observation that if $u \in S_v$, then $v \in T_u$, the following lemma shows that for most vertices, the partition is indeed approximately balanced.

▶ **Lemma 6.** *For any* n*-vertex graph, there are* $> \frac{n}{2}$ *vertices* v *such that* $\frac{|S_v|}{8} \leq |T_v| \leq 8 \cdot |S_v|$*.*

More generally, for any $U \subseteq V$*, there are more than* $\frac{|U|}{2}$ *vertices* $v \in U$ *such that* $\frac{|S_v \cap U|}{8} \leq |T_v \cap U| \leq 8|S_v \cap U|$*.*

Lemma 6 is proved in the full version [16]. Next, we describe how we use Lemma 6 to recursively construct a balanced partition of the vertices into a given number of of sets.

▶ **Lemma 7.** *Given an* n*-vertex graph* $G = (V, E)$ *and a constant* $c > 0$*, in* $\tilde{O}(mn^{1-c})$ *time one can split* V *into disjoint sets* $W, V_1, V_2, \ldots, V_{q+1}$*, where* $q = |W| = n^{1-c}$*, such that with high probability:*

1. *for all* i*,* $|V_i| = \Theta(\frac{n}{q})$*;*
2. *for all* $i \neq j$*, there exists* $w \in W$ *such that either* $V_i \subseteq S_w$,$V_j \subseteq T_w$*, or* $V_i \subseteq T_w$,$V_j \subseteq S_w$*;*
3. *for all* $U \subseteq W$*, let* $V_U = \left(\bigcap_{w \in U} S_w\right) \bigcap \left(\bigcap_{w \in W \setminus U} T_w\right)$*, then* $V_U \subseteq V_i$ *for some* $i \in [q+1]$*.*

Proof. We begin with $W = \emptyset$ and we will iteratively populate W with vertices. We let $\mathcal{V}_0 = \{V\}$ and for all $i \in [q]$ when we add the i^{th} vertex to W, we will construct $\mathcal{V}_i$ from $\mathcal{V}_{i-1}$ by partitioning the largest set in $\mathcal{V}_{i-1}$ into two parts. After adding q vertices to W we will have constructed $\mathcal{V}_q = \{V_1 \ldots V_{q+1}\}$.

For all $i \in [q]$, let A_i, B_i be the largest and smallest sets in $\mathcal{V}_i$, respectively.

We describe how to construct W and $\mathcal{V}_q$ inductively. Suppose $|W| = r - 1$ and we have constructed $\mathcal{V}_{r-1}$. By Lemma 6, if we randomly sample $O(\log^2 n)$ vertices from A_{r-1}, with probability at least $1 - 2^{-\log^2 n} = 1 - n^{-\log n}$ we will sample a vertex w_r such that $A_S = A_{r-1} \cap S_{w_r}$ and $A_T = A_{r-1} \cap T_{w_r}$ differ by a factor of at most 8. We add w_r to W and let $\mathcal{V}_r = \mathcal{V}_{r-1} \cup \{A_S, A_T\} \setminus \{A_{r-1}\}$.

By union bound over the $q = n^{1-c}$ partitionings, with probability at least $1 - n^{1-c-\log n}$, every partitioning produces two sets that differ in size by a factor of at most 8.

We prove property 1 by induction on $|W| = r$. Specifically, we will show that for all $r \in [q]$, $|A_r| \leq 9|B_r|$. This implies that $|A_q| = O(|B_q|)$, and property 1 follows. Lemma 6 implies that $|A_1| \leq 9|B_1|$. Assume inductively that $|A_{r-1}| \leq 9|B_{r-1}|$. Since no subset grows in size, $|A_r| \leq |A_{r-1}|$ and $|B_r| \leq |B_{r-1}|$. If $|B_r| = |B_{r-1}|$, then $|A_r| \leq |A_{r-1}| \leq 9|B_{r-1}| = 9|B_r|$. Otherwise, $|B_r| < |B_{r-1}|$, which implies that B_r is one of the two sets obtained by partitioning A_{r-1}. Then by Lemma 6, $|A_{r-1}| \leq 9|B_r|$. Hence $|A_r| \leq |A_{r-1}| \leq 9|B_r|$, completing the induction.

Property 2 follows from the partitioning procedure: for any $i \neq j$, if for all $w \in W$, $V_i, V_j \subseteq S_w$ or $V_i, V_j \subseteq T_w$ then $V_i \cup V_j$ would never have been partitioned.

Property 3 also follows from the partitioning procedure: observe that for all $w \in W$ and all $U \subseteq W$, $V_U \subseteq S_w$ or $V_U \subseteq T_w$, so V_U is never partitioned and thus $V_U \subseteq V_i$ for some $i \in [q+1]$.

Since we sample $n^{1-c}\log^2 n$ vertices and for all v finding S_v, T_v takes $O(m)$ time, the runtime is $\tilde{O}(mn^{1-c})$. ◀

4 Min-Diameter Algorithm

Throughout this section, let D be the min-diameter, and let s^*, t^* the endpoints of the min-diameter. In this section we prove the time/accuracy trade-off theorem for Min-Diameter.

▶ **Theorem 8.** *For any integer $0 < \ell \leq O(\log n)$, there is an $\tilde{O}(mn^{1/(\ell+1)})$ time randomized algorithm that, given a directed weighted graph G with edge weights non-negative and polynomial in n, can output an estimate $\tilde{D}$ such that $D/(4\ell - 1) \leq \tilde{D} \leq D$ with high probability, where D is the min-diameter of G.*

We first prove a special case of Theorem 8 where $\ell = 1$, and the rest of the proof can be found in the full version [16].

4.1 An $\tilde{O}(m\sqrt{n})$ time 3-approximation

▶ **Theorem 9** (Theorem 8 with $\ell = 1$)**.** *There is an $\tilde{O}(m\sqrt{n})$ time randomized algorithm, that given a directed weighted graph $G = (V, E)$ with edge weights non-negative and polynomial in n, can output an estimate $\tilde{D}$ such that $D/3 \leq \tilde{D} \leq D$ with high probability, where D is the min-diameter of G.*

4.1.1 Algorithm Description

Applying Lemma 7 with $q = \sqrt{n}$ we obtain a partition of the vertices into $W, V_1, V_2, \ldots, V_{\sqrt{n}+1}$.

We perform Dijkstra's algorithm from every vertex in W and define $D' = \max_{w \in W} \varepsilon(w)$. We will later show that D' is a good approximation of the Min-Diameter when s^* and t^* are not in the same vertex set V_i.

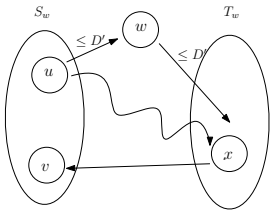

Figure 1 The case where $u, v \in S_w$ and the shortest path from u to v contains a node $x \in T_w \cup \{w\}$.

For every $i \in [\sqrt{n}+1]$, define $W_i^S = \{w \in W : V_i \subseteq S_w\}$, and $W_i^T = \{w \in W : V_i \subseteq T_w\}$. Then, for every i, we construct two graphs G_i^S and G_i^T. The first graph G_i^S contains all vertices of V_i and an additional node w_i^S. It has the following edges:

1. For every directed edge $(u, v) \in E$ such that $u, v \in V_i$, add this edge to G_i^S.
2. Add a directed edge from w_i^S to every $v \in V_i$, with weight $\max\left\{\min_{w \in W_i^S} d(w, v) - D', 0\right\}$, and a directed edge from every $v \in V_i$ to w_i^S with weight 0.

The second graph G_i^T is symmetric to G_i^S. It contains all vertices in V_i and an additional node w_i^T. It has the following edges:

1. For every directed edge $(u, v) \in E$ such that $u, v \in V_i$, add this edge to G_i^T.
2. Add a directed edge from every $v \in V_i$ to w_i^T, with weight $\max\left\{\min_{w \in W_i^T} d(v, w) - D', 0\right\}$, and add a directed edge from w_i^T to every $v \in V_i$ with weight 0.

For all i, we run an exact all-pairs shortest paths algorithm on G_i^S and G_i^T. This allows us to compute for all i and all $u, v \in V_i$ the quantity $\min\{d_{G_i^S}(u, v), d_{G_i^T}(u, v)\}$, which we denote by $d_i'(u, v)$.

We choose the larger between D' and $\max_{i \in [\sqrt{n}+1], u, v \in V_i} \min\{d_i'(u, v), d_i'(v, u)\}$ as our final estimate for the min-diameter.

4.1.2 Analysis

The following lemma will be used to show that D' is a good estimate for the min-diameter if s^* and t^* happen to fall into different sets V_i

▶ **Lemma 10.** *For all vertices v, if either $s^* \in S_v$, $t^* \in T_v$, or $t^* \in S_v$, $s^* \in T_v$, then $\varepsilon(v) \geq D/2$.*

Proof. We only consider the case when $s^* \in S_v$ and $t^* \in T_v$ as the other case is symmetric. By way of contradiction, assume that $\varepsilon(v) < D/2$, then we have $d_{\min}(s^*, v) < D/2$ and $d_{\min}(t^*, v) < D/2$. Since $s^* \in S_v$, $d(s^*, v) = d_{\min}(s^*, v) < D/2$; similarly, since $t^* \in T_v$, $d(v, t^*) = d_{min}(t^*, v) < D/2$. Therefore, by the triangle inequality, $d(s^*, t^*) < D$, a contradiction. ◀

The next two lemmas are used for the case where s^* and t^* fall into the same set V_i.

▶ **Lemma 11.** *For every i, and every pair of vertices $u, v \in V_i$, $d_i'(u, v) \leq d(u, v)$; that is, $\min\{d_{G_i^S}(u, v), d_{G_i^T}(u, v)\} \leq d(u, v)$.*

Proof. Take any shortest path in the original graph G from u to v. If this path does not leave V_i, then this path also exists in G_i^S and G_i^T, and thus the inequality is true.

It remains to prove for the case when the shortest u, v path in the original graph leaves V_i. Let $x \notin V_i$ be any vertex on a shortest u, v path. By Lemma 7, property 2, there exists $w \in W$ such that $x \in S_w \cup \{w\}$ and $V_i \subseteq T_w$, or $x \in T_w \cup \{w\}$ and $V_i \subseteq S_w$. We first assume $x \in T_w \cup \{w\}$ and $V_i \subseteq S_w$ as shown in Figure 1, and the other case is symmetric.

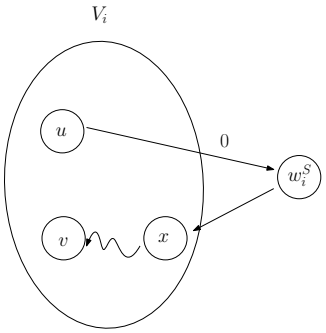

Figure 2 A shortest u, v path in G_i^S that contains w_i^S. The path goes from u, directly to w_i^S using a weight 0 edge, then directly to a vertex x, and finally reaches v.

Since x is on the shortest path from u to v, we have $d(u, v) \geq d(x, v)$. Also, we have $d(w, x) \leq D'$, by definition of D'. Therefore,

$$d(u, v) \geq d(x, v) \geq d(x, v) + (d(w, x) - D') \geq d(w, v) - D'. \tag{1}$$

Now consider the path $u \to w_i^S \to v$ in G_i^S. The first part $u \to w_i^S$ costs 0, because there is an edge from u to w_i^S with weight 0; the second part $w_i^S \to v$ costs at most $\max\{0, d(w, v) - D'\}$. If $d(w, v) < D'$, then $d'_i(u, v) \leq d_{G_i^S}(u, v) = 0 \leq d(u, v)$; otherwise, $d'_i(u, v) \leq d_{G_i^S}(u, v) \leq d(w, v) - D' \leq d(u, v)$, where the last step is Equation 1.

When $x \in S_w \cup \{w\}$, and $V_i \subseteq T_w$, we have a symmetric argument: $d(u, v) \geq d(u, x) \geq d(u, x) + (d(x, w) - D') \geq d(u, w) - D'$. Consider the path $u \to w_i^T \to v$ in G_i^T. The second part $w_i^T \to v$ costs 0, because there is an edge from w_i^T to v with weight 0; the first part $u \to w_i^T$ costs at most $\max\{0, d(u, w) - D'\}$. If $d(u, w) < D'$, then $d'_i(u, v) \leq d_{G_i^T}(u, v) = 0 \leq d(u, v)$; otherwise, $d'_i(u, v) \leq d_{G_i^T}(u, v) \leq d(u, w) - D' \leq d(u, v)$. ◀

▶ **Lemma 12.** *For every i, and every pair of vertices $u, v \in V_i$, $d'_i(u, v) \geq d(u, v) - 2D'$; that is, $d_{G_i^S}(u, v) \geq d(u, v) - 2D'$ and $d_{G_i^T}(u, v) \geq d(u, v) - 2D'$.*

Proof. We only provide full proof for $d_{G_i^S}(u, v) \geq d(u, v) - 2D'$. The inequality for G_i^T can be proved by a symmetrical argument. If the shortest path from u to v in G_i^S does not contain w_i^S, then this path also exists in the original graph G, and thus the inequality is true.

Otherwise, the shortest path from u to v in G_i^S contains w_i^S, as shown in Figure 2. All edges on the shortest path from w_i^S to v exist in the original graph G except for the first edge from w_i^S to some node x, since a shortest path cannot use the vertex w_i^S more than once. That is, $d_{G_i^S}(x, v) = d(x, v)$.

By the definition of w_i^S and the edges incident to it, there exists a $w \in W_i^S$ such that $d(w, x) \leq d_{G_i^S}(w_i^S, x) + D'$. Thus, we have

$$\begin{aligned}
d_{G_i^S}(u, v) &= d_{G_i^S}(u, w_i^S) + d_{G_i^S}(w_i^S, x) + d_{G_i^S}(x, v) \\
&= d_{G_i^S}(w_i^S, x) + d_{G_i^S}(x, v) && \text{since } d_{G_i^S}(u, w_i^S) = 0 \text{ by construction} \\
&= d_{G_i^S}(w_i^S, x) + d(x, v) && \text{from argument above} \\
&\geq d(w, x) - D' + d(x, v) && \text{by the definition of } w \\
&\geq d(w, v) - D' && \text{by the triangle inequality} \\
&\geq (d(w, v) - D') + (d(u, w) - D') && \text{since } d(u, w) \leq D' \text{ by definition} \\
&\geq d(u, v) - 2D' && \text{by the triangle inequality}
\end{aligned}$$

◀

We are now ready to prove our approximation ratio guarantee: $D/3 \leq \tilde{D} \leq D$. Clearly $D' \leq D$ because D' is the min-eccentricity of a vertex. By Lemma 11 $\max_{i,u\in V_i,v\in V_i} \min\{d'_i(u,v), d'_i(v,u)\} \leq \max_{i,u\in V_i,v\in V_i} d_{min}(u,v) \leq D$. Therefore, we never over estimate the Min-Diameter.

If $s^* \in W$ or $t^* \in W$, then since we run Dijkstra from all vertices in W we have $D' = D$. So assuming that $s^*, t^* \notin W$, we have two cases.

Case 1: s^* and t^* are not in the same vertex set V_i. By Lemma 7, property 2, there exists $w \in W$ such that one of s^* and t^* is in S_w and the other is in T_w, so by Lemma 10, $\varepsilon(w) \geq D/2$. Since $D' \geq \varepsilon(w)$, we have $D' \geq D/2$.

Case 2: s^* and t^* are in the same vertex set V_i for some i. By Lemma 12, $\min(d'_i(s^*,t^*), d'_i(t^*,s^*)) \geq d_{min}(s^*,t^*) - 2D' = D - 2D'$. Since $\max\{D-2D', D'\} \geq D/3$, we get a 3-approximation.

Runtime analysis

It takes $\tilde{O}(m\sqrt{n})$ time to perform the partitioning from Lemma 7 and to perform Dijkstra's algorithm from all $w \in W$ since $|W| = O(\sqrt{n})$.

For all i, the number of vertices in G_i^S is $|V_i| + 1 = O(\sqrt{n})$ with high probability by property 1 of Lemma 7 and the number of edges is $m_i + O(\sqrt{n})$ where m_i is the number of edges in the graph induced by V_i. Hence we can run an all-pairs shortest paths algorithm on G_i^S in time $\tilde{O}((m_i + \sqrt{n})\sqrt{n})$. Summing over all i gives us $\tilde{O}(m\sqrt{n})$. The same analysis also works for G_T^i.

References

1. 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.
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. 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.
4. Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 267–280, 2018.
5. B. Ben-Moshe, B. K. Bhattacharya, Q. Shi, and A. Tamir. Efficient algorithms for center problems in cactus networks. *Theoretical Computer Science*, 378(3):237–252, 2007.
6. P. Berman and S. P. Kasiviswanathan. Faster Approximation of Distances in Graphs. In *Proc. WADS*, pages 541–552, 2007.
7. 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.
8. 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.

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

10 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.

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

12 V. Chepoi and F. F. Dragan. A Linear-Time Algorithm for Finding a Central Vertex of a Chordal Graph. In *ESA*, pages 159–170, 1994.

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

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

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

16 Mina Dalirrooyfard, Virginia Vassilevska Williams, Nikhil Vyas, Nicole Wein, Yinzhan Xu, and Yuancheng Yu. Approximation Algorithms for Min-Distance Problems, 2019. `arXiv:1904.11606`.

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

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

19 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.

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

21 Seth Pettie. A faster all-pairs shortest path algorithm for real-weighted sparse graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 85–97. Springer, 2002.

22 Seth Pettie and Vijaya Ramachandran. A Shortest Path Algorithm for Real-Weighted Undirected Graphs. *SIAM J. Comput.*, 34(6):1398–1431, 2005. `doi:10.1137/S0097539702419650`.

23 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, New York, NY, USA, 2013. ACM. `doi:10.1145/2488608.2488673`.

24 O. Weimann and R. Yuster. Approximating the Diameter of Planar Graphs in Near Linear Time. In *Proc. ICALP*, 2013.

25 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.

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

27 Raphael Yuster. Computing the diameter polynomially faster than APSP. *arXiv preprint*, 2010. `arXiv:1011.6181`.