Constant Approximation of Min-Distances in Near-Linear Time

Shiri Chechik
Blavatnik School of Computer Science
Tel Aviv University
Tel Aviv, Israel
shiri.chechik@gmail.com

Tianyi Zhang
Blavatnik School of Computer Science
Tel Aviv University
Tel Aviv, Israel
tianyiz21@tauex.tau.ac.il

***Abstract*—In a weighed directed graph $G = (V, E, \omega)$ with m edges and n vertices, we are interested in its basic graph parameters such as diameter, radius and eccentricities, under the nonstandard measure of min-distance which is defined for every pair of vertices $u, v \in V$ as the minimum of the shortest path distances from u to v and from v to u. Similar to standard shortest paths distances, computing graph parameters exactly in terms of min-distances essentially requires $\tilde{\Omega}(mn)$ time under plausible hardness conjectures [1]. Hence, for faster running time complexities we have to tolerate approximations.**

Abboud, Vassilevska Williams and Wang [SODA 2016] were the first to study min-distance problems, and they obtained constant factor approximation algorithms in acyclic graphs, with running time $\tilde{O}(m)$ and $\tilde{O}(m\sqrt{n})$ for diameter and radius, respectively. The time complexity of radius in acyclic graphs was recently improved to $\tilde{O}(m)$ by Dalirrooyfard and Kaufmann [ICALP 2021], but at the cost of an $O(\log n)$ approximation ratio. For general graphs, the authors of [DWV+, ICALP 2019] gave the first constant factor approximation algorithm for diameter, radius and eccentricities which runs in time $\tilde{O}(m\sqrt{n})$; besides, for the diameter problem, the running time can be improved to $\tilde{O}(m)$ while blowing up the approximation ratio to $O(\log n)$.

A natural question is whether constant approximation and near-linear time can be achieved simultaneously for diameter, radius and eccentricities; so far this is only possible for diameter in the restricted setting of acyclic graphs. In this paper, we answer this question in the affirmative by presenting near-linear time algorithms for all three parameters in general graphs.

I. Introduction

Diameter, radius and eccentricities are basic graph parameters that have been widely studied (e.g., [1]–[9], [12]–[17]). These three parameters are related in the following sense: the eccentricity of any vertex in a graph is the longest distance between this vertex and any other vertex, while diameter and radius are the maximum and the minimum eccentricity of all vertices, respectively.

There are several different ways to define the notion of distances with which diameter, radius and eccentricities are instantiated. In an undirected graph, the traditional sense of distances refers to the length of the shortest path $\mathbf{dist}(\cdot, \cdot)$ between two vertices. In directed graphs, for any pair of vertices u and v, there are various ways to define distances in terms of shortest paths, other than the $\mathbf{dist}(u, v)$. These include round-trip distances [9] which is defined as $\mathbf{dist}(u, v) + \mathbf{dist}(v, u)$, and max-distances [2] which is $\max\{\mathbf{dist}(u, v), \mathbf{dist}(v, u)\}$, and min-distances [2] which is $\min\{\mathbf{dist}(u, v), \mathbf{dist}(v, u)\}$. In this work we focus on the notion of min-distances, and under this specific distance measure, the corresponding graph parameters are called min-diameter, min-radius, min-eccentricities for short. The min distance is a natural definition of distance with clear applications, for example, in a medical emergency one needs to decide if to call an ambulance (from the hospital) or to get to the hospital.

As with other standard definitions of distances, computing these graph parameters exactly under min-distances cannot be done in [2]$O(m^{2-\epsilon})$ time for any constant $\epsilon > 0$ [2], [14], so it is natural to resort to approximation algorithms. In [2], the authors provided a near-linear time 2-approximation of min-diameter in acyclic graphs, and a 3-approximation algorithm for min-radius that runs in time $\tilde{O}(mn^{1/2})$ in an n-vertex m-edge acyclic graphs; also, if only near-linear running time is allowed, then their algorithms can decide if the min-radius of the input graph is finite. This approximation ratio of 3 was improved to 2 in a recent paper by Dalirrooyfard and Kaufmann [10], which is conditionally tight for subquadratic time algorithms; more generally, their algorithm computes k-approximations of min-radius and min-eccentricities in time $\tilde{O}(\min\{mn^{1/k}, m^{2^{k-1}/(2^k-1)}n\})$ for any integer $k \geq 2$; in particular, in the near-linear time regime, their algorithm has $O(\log n)$ approximation. On the conditional lower bound side, it was shown in [2], [10] that $(1.5-\epsilon)$ approximation of min-diameter or $(2-\epsilon)$-approximation of min-radius in acyclic graphs requires $\max\{m^{2-o(1)}, n^{\omega-o(1)}\}$ time.

All the above upper bounds only hold in acyclic graphs. For general graphs, the authors of [11] designed 3-approximation algorithms for min-diameter and min-radius general graphs with running time $\tilde{O}(mn^{1/2})$; their algorithm can also be adapted to compute min-eccentricities of all vertices with the same running time, at the cost of a larger approximation ratio of $5 + \epsilon$. Additionally, they also show a trade-off between time and approximation ratio in the case of min-diameter: for

This publication is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 803118 UncertainENV).

[1]As usual, the $\tilde{O}(\cdot)$ notation hides poly-logarithmic factors in n

[2]Conventionally, assume the input graph has n vertices and m edges.

2575-8454/22/$31.00 ©2022 IEEE
DOI 10.1109/FOCS54457.2022.00089

any integer $2 \leq k \leq O(\log n)$, a $(4k-5)$-approximation of min-diameter can be computed in time $\tilde{O}(mn^{1/k})$; in particular, near-linear time algorithms can achieve $O(\log n)$-approximations. Unfortunately, unlike in acyclic graphs, so far we do not have any non-trivial near-linear time approximation algorithms for min-radius or min-eccentricities in general graphs, even if $O(\log n)$ approximations are allowed.

Following this line of works, we would to like to highlight a natural question whether constant approximation and near-linear time can be achieved at the same time in general graphs for min-diameter, min-radius and min-eccentricities.

A. Our results

As our main result, we show that constant-factor approximation of all three graph parameters in general graphs can be computed in near-linear time. Our algorithms for min-radius and min-eccentricities are closely related, while the algorithm for min-diameter is more different, so we formulate our results in two separate statements below.

Theorem 1.1: There is a randomized algorithm that computes a 4-approximation of min-diameter of G in $\tilde{O}(m)$ time.

Theorem 1.2: There is a randomized algorithm that computes a 4-approximation of min-radius of G in $\tilde{O}(m)$ time. Also, for any constant $\epsilon > 0$, there is a randomized algorithm that computes $(5+\epsilon)$-approximations of min-eccentricities of G in $\tilde{O}(m/\epsilon)$ time.

As our secondary result, we also restrict ourselves to directed acyclic graphs, and achieve 3-approximations of min-radius and min-eccentricities in near-linear time. Previously, 3-approximations of min-radius and min-eccentricities needed $\tilde{O}(\min\{mn^{1/3}, m^{4/7}n\})$ running time [2], [10], and near-linear algorithms could only obtain $O(\log n)$-approximations [10]. For a detailed comparison to previous works, please check Table I.

Theorem 1.3: There is a deterministic algorithm that computes a 3-approximation of min-radius of directed acyclic graph G in $\tilde{O}(m)$ time. Furthermore, for any constant $\epsilon > 0$, there is a deterministic algorithm that computes $(3+\epsilon)$-approximations of min-eccentricities of G in $\tilde{O}(m/\epsilon)$ time.

B. Technical overview

In this subsection, let us describe an outline of our algorithm for min-diameter and min-radius in general graphs.

min-diameter: Given a threshold D, we want to either find a pair of vertices whose min-distance in G is at least $D/4$, or prove that the min-diameter of G is less than D; if it is done, then to approximate the min-diameter in G, we can apply a binary search on the best choice of D. To do this, we devise a recursive algorithm $\mathsf{MinDiameter}(S, C)$ whose inputs are two sets of vertices $C \subseteq S \subseteq V$; at the beginning, we simply call $\mathsf{MinDiameter}(V, V)$. During the recursion, set C represents a set of candidate endpoints of the min-diameter in subgraph $G[S]$.

The starting point of the algorithm is sampling a random vertex t from C and computing single source shortest paths in $G[S]$ to and from t, and defining two sets $U_1 = \{u \in C \mid \mathbf{dist}_{G[S]}(t,u) < D/4\}$ and $U_2 = \{u \in C \mid \mathbf{dist}_{G[S]}(u,t) < D/4\}$ accordingly. We can assume $U_1 \cup U_2 = C$, since otherwise we can already find a vertex z such that $\mathbf{dist}^{\min}_{G[S]}(t,z) \geq D/4$. A basic observations is that, if there exists a pair $x, y \in S$ whose min-distance is at least D, then x, y belong to the same set $U_b, b \in \{1,2\}$. Therefore, a natural strategy is to recurse on $(S_1 = U_1, U_1)$ and $(S_2 = U_2, U_2)$ to detect vertex pairs with large min-distances.

There are two main concerns with this approach. First, U_1 and U_2 can be intersecting in general, so recursing on both $G[U_1]$ and $G[U_2]$ might be too costly. A second concern with this approach is **soundness**: even if a recursion finds a pair of vertices such that $\mathbf{dist}^{\min}_{G[U_1]}(x,y) \geq D/4$, it is still possible that $\mathbf{dist}^{\min}_{G[S]}(x,y) < D/4$.

To handle the second concern, consider any pair of vertices $x, y \in U_1$ such that $\mathbf{dist}^{\min}_{G[S]}(x,y) < D/4$, say $\mathbf{dist}_{G[S]}(x,y) < D/4$. Then by definition of U_1, it implies that $\mathbf{dist}_{G[S]}(t,y) \leq \mathbf{dist}_{G[S]}(t,x) + \mathbf{dist}_{G[S]}(x,y) < D/2$. So, if we include all vertices from $\mathbf{B}^+_{G[S]}(t, D/2)$ in S_1 (symmetrically, all vertices $\mathbf{B}^-_{G[S]}(t, D/2)$ in S_2), then soundness can be guaranteed; here $\mathbf{B}^+_{G[S]}(t, D/2)$ and $\mathbf{B}^-_{G[S]}(t, D/2)$ are out/in balls centered at t with radius $D/2$ in $G[S]$. So recursing on $(S_1 = \mathbf{B}^+_{G[S]}(t, D/2), U_1)$ and $(S_2 = \mathbf{B}^-_{G[S]}(t, D/2), U_2)$ indeed ensures correctness. However, S_1 and S_2 could still be intersecting; namely it could be the case that

$$S_1 \cap S_2 = \mathbf{B}^+_{G[S]}(t, D/2) \cap \mathbf{B}^-_{G[S]}(t, D/2) \neq \emptyset$$

So, for the running efficiency, we may want to set $S_1 = \mathbf{B}^+_{G[S]}(t, D/2) \setminus \mathbf{B}^-_{G[S]}(t, D/2)$ and $S_2 = \mathbf{B}^-_{G[S]}(t, D/2) \setminus \mathbf{B}^+_{G[S]}(t, D/2)$. It is still possible that for some pairs $x, y \in S_1$, $\mathbf{dist}_{G[S]}(x,y) < D/4$, but the shortest path goes outside of S_1 and intersects $\mathbf{B}^-_{G[S]}(t, D/2)$. In this case, we have $\mathbf{dist}_{G[S]}(x,t) < 3D/4$. Fortunately, we do not need to worry about such kind of x, since for any pairs $u, v \in S_1$ of true interest, namely $\mathbf{dist}^{\min}_{G[S]}(u,v) \geq D$, we have $\mathbf{dist}_{G[S]}(u,t), \mathbf{dist}_{G[S]}(v,t) \geq 3D/4$. So we can safely recurse on inputs $(S_1, C_1), (S_2, C_2)$, where

$$C_1 = U_1 \cap \{u \mid \mathbf{dist}_{G[S]}(u,t) \geq 3D/4\}$$
$$C_2 = U_2 \cap \{u \mid \mathbf{dist}_{G[S]}(t,u) \geq 3D/4\}$$

while soundness is preserved along the way.

To bound the recursion depth, we apply an important lemma from [11] to argue that if t is picked uniformly at random then with constant probability both C_1 and C_2 are at most constant fraction of C. By Chernoff bound we show later that it follows that the recursion depth is $O(\log n)$ with high probability.

min-radius: Let us begin with a near-linear time algorithm that gives a $O(\log n)$ approximation of min-radius. Given a threshold R, we want to either find a vertex from V whose min-eccentricity in G is at most $O(R \log n)$, or prove that the min-radius of G is larger than R. The key subroutine $\mathrm{LogRadius}$ is a recursive subroutine that accepts two parameters (H, C), where H is a contraction graph of G, and

TABLE I
A COMPARISON AGAINST PREVIOUS WORKS.

problem setting	running time	approximation	reference
min-diameter in acyclic graphs	$\tilde{O}(m)$	2	[2]
min-diameter in general graphs	$\tilde{O}(mn^{1/k})$	$4k-5, \forall k \geq 2$	[11]
	$\tilde{O}(m)$	4	new
min-radius in general graphs	$\tilde{O}(mn^{1/2})$	3	[11]
	$\tilde{O}(m)$	4	new
min-ecc. in general graphs	$\tilde{O}(mn^{1/2}/\epsilon)$	$5+\epsilon$	[11]
	$\tilde{O}(m/\epsilon)$	$5+\epsilon$	new
min-radius in acyclic graphs	$\tilde{O}(mn^{1/2})$	3	[2]
	$\tilde{O}(\min\{mn^{1/k}, m^{2^{k-1}/(2^k-1)}n\})$	$k, \forall k \geq 2$	[10]
	$\tilde{O}(m)$	3	new
min-ecc. in acyclic graphs	$\tilde{O}(\min\{mn^{1/k}, m^{2^{k-1}/(2^k-1)}n\}/\epsilon)$	$k+\epsilon, \forall k \geq 2$	[10]
	$\tilde{O}(m/\epsilon)$	$3+\epsilon$	new

$C \subseteq V \cap V(H)$ is a vertex subset which does not include any contracted nodes.

The main algorithm simply runs $\mathrm{LogRadius}(G, V)$. Consider an arbitrary level of recursion where $C \neq \emptyset$. The algorithm starts by taking a random source vertex $t \in C$ and compute single-source shortest paths to and from t in H. If $V(H) = \mathbf{B}^+_H(t, 2R) \cup \mathbf{B}^-_H(t, 2R)$, then we will return t as an $O(R \log n)$-min-center of G; here an $O(R \log n)$-min-center means a vertex whose eccentricity is bounded by $O(R \log n)$. Otherwise, let us assume $V(H) \neq \mathbf{B}^+_H(t, 2R) \cup \mathbf{B}^-_H(t, 2R)$.

Define $W = V \setminus (\mathbf{B}^+_H(t, 2R) \cup \mathbf{B}^-_H(t, 2R))$. If the min-radius of H is at most R, then the min-center must be in $\mathbf{B}^+_H(t, R) \cup \mathbf{B}^-_H(t, R)$. Our strategy is to search for the min-center in sets $C_1 = C \cap \mathbf{B}^+_H(t, R)$ and $C_2 = C \cap \mathbf{B}^-_H(t, R)$ recursively on two smaller graphs. Construct two graphs as following: contract all vertices in $\mathbf{B}^-_H(t, 2R)$ into a single node and call the new graph H_1; symmetrically we can define a new graph H_2. After that, invoke two recursive calls $\mathrm{LogRadius}(H_1, C_1)$ and $\mathrm{LogRadius}(H_2, C_2)$. For the approximation ratio, as each contracted node represents a subgraph with radius at most $2R$, the eccentricity of the final output vertex can be bounded by $O(R \cdot (\text{depth of recursion}))$.

To bound the recursion depth by $O(\log n)$, we again use the lemma from [11] that shows with probability at least $1/2$ over the choice of $t \in C$, we have $\max\{|C_1|, |C_2|\} \leq \frac{8}{9}|C|$. Therefore, with high probability, the recursion depth of $\mathcal{T}$ is bounded by $O(\log n)$. However, it is not enough to bound the total running time, since at each recursion node (H, C), the subgraphs H_1 and H_2 may intersect by a lot, since $W \subseteq V(H_1) \cap V(H_2)$. The key observation is that, for any vertex v, if it once belongs to the set $W \subseteq V(H_1) \cup V(H_2)$ and is inherited by both subgraphs H_1 and H_2, then v can never belong to both children of any nodes (H', C') which is a descendant of either (H_1, C_1) or (H_2, C_2). Therefore, all the tree nodes that contain v are a union of at most two tree paths starting at root. As the depth of $\mathcal{T}$ is bounded by $O(\log n)$, the total number of times that v appears in any contracted graph H is also bounded by $O(\log n)$.

To improve the approximation ratio from $\log n$ to constant, instead of recurring on contracted graphs H_1, H_2 of H, we carefully recurse on two induced subgraphs of H. More specifically, we will divide C into two sets $C = C_1 \cup C_2$, and then recurse on instances (H_1, C_1) and (H_2, C_2), respectively. The total running time will be bounded as near-linear using similar techniques we just discussed in the previous paragraph. However, another key technical difficulty of such kind of recursions is that these subgraphs need to preserve exact shortest paths in H. Clearly, this task is impossible if we wish to preserve all-pairs shortest paths. The idea is to preserve shortest paths in subgraphs H_1, H_2 only between vertices that are not yet known to be close in min-distances. More concretely, we will introduce a third input parameter $T \subseteq V$ to the recursive procedure, such that for any pairs of vertices $c \in C, v \in V \setminus T$, we can guarantee that $\mathbf{dist}^{\min}_G(c, v) = O(R)$. So, in subgraph H, we only need to focus on min-distances between $c \in C$ and $v \in T$. When building the two subgraphs H_1, H_2, we will divide $T = T_1 \cup T_2$ and somehow ensure that pairwise distances between $c \in C_b$ and $v \in T_b \cup C_b$ will be preserved exactly in H_b, $\forall b \in \{1, 2\}$. After that, we will recurse on instances (H_1, T_1, C_1) and (H_2, T_2, C_2), respectively.

II. PRELIMINARIES

Let $G = (V, E, \omega)$ be a directed graph with positive polynomially-bounded integral edge weights on n vertices and m edges. For any directed path π in G, let $\omega(\pi)$ be the total weight of this path. For any $S \subseteq V$, let $G[S]$ be the induced subgraph of S. For any positive value r and vertex $u \in V$, define $\mathbf{B}^+_{G[S]}(u, r) = \{v \mid \mathbf{dist}_{G[S]}(u, v) \leq r\}$ and $\mathbf{B}^-_{G[S]}(u, r) = \{v \mid \mathbf{dist}_{G[S]}(v, u) \leq r\}$; the subscript G is omitted when $S = V$. For two different induced subgraphs H_1, H_2 of G, $H_1 \setminus H_2$ refers to the induced subgraph on vertices $V(H_1) \setminus V(H_2)$.

For any pair of vertices $u, v \in V$, the min-distance between u, v is defined as $\mathbf{dist}^{\min}_G(u, v) = \min\{\mathbf{dist}_G(u, v), \mathbf{dist}_G(v, u)\}$; the subscript G is usually omitted when it can be deduced from context. A vertex $c \in V$ is called an **r-min-center** if its min-eccentricity is bounded by r, namely $\max_{v \in V} \mathbf{dist}^{\min}(c, v) \leq r$.

For each vertex $u \in S$ where $S \subseteq V$, define $X_{G[S]}(u) = \{v \neq u \mid \mathbf{dist}_{G[S]}(u, v) < \mathbf{dist}_{G[S]}(v, u)\}$, and $Y_{G[S]}(u) =$

$S \setminus (X_{G[S]}(u) \cup \{u\})$. The subscript $G[S]$ is usually omitted when it is clear from context. Our algorithms will use a powerful lemma by [11].

Lemma 2.1 ([11]): For any subset $U \subseteq S$, there are more than $|U|/2$ vertices $u \in U$ such that $|X_{G[S]}(u) \cap U|/8 \leq |Y_{G[S]}(u) \cap U| \leq 8|X_{G[S]}(u) \cap U|$.

III. Min-diameter in general graphs

Given a threshold D, we want to either find a pair of vertices whose min-distance in G is at least $D/4$, or prove that the min-diameter of G is less than D; if it is done, then to approximate the min-diameter in G, we can apply a binary search on the best choice of D. The main algorithm is a recursive procedure $\mathsf{MinDiameter}(S, C)$, where $C \subseteq S \subseteq V$ are two subsets. At the root of the recursion, run $\mathsf{MinDiameter}(V, V)$.

For general inputs of S, C, procedure MinDiameter works as follows. It first picks a vertex t in C uniformly at random and computes single source shortest paths in $G[S]$ to and from t. Then, compute sets $U_1 = \{u \in C \mid \mathbf{dist}_{G[S]}(t, u) < D/4\}$ and $U_2 = \{u \in C \mid \mathbf{dist}_{G[S]}(u, t) < D/4\}$. If $U_1 \cup U_2 \neq S$ then the algorithm picks a vertex $z \in C \setminus (U_1 \cup U_2)$ and returns $\{t, z\}$.

Otherwise, compute $S_1 = \mathbf{B}^+_{G[S]}(t, D/2) \setminus \mathbf{B}^-_{G[S]}(t, D/2)$ and $S_2 = \mathbf{B}^-_{G[S]}(t, D/2) \setminus \mathbf{B}^+_{G[S]}(t, D/2)$, and $C_1 = U_1 \cap \{u \mid \mathbf{dist}_{G[S]}(u, t) \geq 3D/4\}$ and $C_2 = U_2 \cap \{u \mid \mathbf{dist}_{G[S]}(t, u) \geq 3D/4\}$. Finally, recurse on inputs (S_1, C_1) and (S_2, C_2). See Algorithm 1 for a formal description.

Algorithm 1: $\mathsf{MinDiameter}(S, C)$

1 **if** $C = \emptyset$ **then**
2 **return null**;
3 sample a vertex $t \in C$ uniformly at randon, and compute SSSP in $G[S]$ to and from t;
4 compute $U_1 = \{u \in C \mid \mathbf{dist}(t, u) < D/4\}$;
5 compute $U_2 = \{u \in C \mid \mathbf{dist}(u, t) < D/4\}$;
6 **if** $U_1 \cup U_2 \neq C$ **then**
7 pick an arbitrary vertex $z \in C \setminus (U_1 \cup U_2)$;
8 **return** (t, z);
9 compute $S_1 = \mathbf{B}^+_{G[S]}(t, D/2) \setminus \mathbf{B}^-_{G[S]}(t, D/2)$ and $S_2 = \mathbf{B}^-_{G[S]}(t, D/2) \setminus \mathbf{B}^+_{G[S]}(t, D/2)$;
10 compute $C_1 = U_1 \cap \{u \mid \mathbf{dist}_{G[S]}(u, t) \geq 3D/4\}$ and $C_2 = U_2 \cap \{u \mid \mathbf{dist}_{G[S]}(t, u) \geq 3D/4\}$;
11 run $(t_1, z_1) \leftarrow \mathsf{MinDiameter}(S_1, C_1)$;
12 run $(t_2, z_2) \leftarrow \mathsf{MinDiameter}(S_2, C_2)$;
13 **return** (t_1, z_1) or (t_2, z_2), whichever is not null;

A. Proof of correctness

Consider any recursion on input (S, C). Assume the algorithm reaches line-9. First, we prove a basic property regarding endpoints of the min-diameter.

Lemma 3.1: Consider any pair of vertices $u, v \in S$ such that $\mathbf{dist}^{\min}_{G[S]}(u, v) \geq D$. Then $u, v \in C_1$ or $u, v \in C_2$.

Proof: If u, v belong to different U_1, U_2 respectively, then

$$\mathbf{dist}_{G[S]}(v, u) \leq \mathbf{dist}_{G[S]}(v, t) + \mathbf{dist}_{G[S]}(t, u) < D/2$$

which is a contradiction. For the rest, let us assume $u, v \in U_1$. Then, by triangle inequality, $\mathbf{dist}_{G[S]}(u, t) \geq \mathbf{dist}_{G[S]}(u, v) - \mathbf{dist}_{G[S]}(t, v) \geq 3D/4$, and so $u \in C_1$. Similarly we can show $v \in C_1$ as well. ∎

The next lemma shows that for any two vertices in C_1, C_2 whose min distance in $G[S_1], G[S_2]$ is at least $D/4$, their distance in $G[S]$ is also at least $D/4$.

Lemma 3.2: For every two vertices $x, y \in C_1$ and $\mathbf{dist}^{\min}_{G[S]}(x, y) < D/4$, we have $\mathbf{dist}^{\min}_{G[S_1]}(x, y) < D/4$. Similarly, for every two vertices $x, y \in C_2$ and $\mathbf{dist}^{\min}_{G[S]}(x, y) < D/4$, we have $\mathbf{dist}^{\min}_{G[S_2]}(x, y) < D/4$.

Proof: By symmetry, we only focus on the first half of the statement. Consider two vertices x and y such that $x, y \in C_1$ and $\mathbf{dist}^{\min}_{G[S]}(x, y) < D/4$. Assume without loss of generality that $\mathbf{dist}_{G[S]}(x, y) \leq \mathbf{dist}_{G[S]}(y, x)$. So $\mathbf{dist}_{G[S]}(x, y) < D/4$. Let ρ be the shortest path from x to y in $G[S]$. We first show that ρ is contained entirely in $\mathbf{B}_{G[S]}(t, D/2)$. In fact, for any vertex $z \in \rho$, $\mathbf{dist}_{G[S]}(t, z) \leq \mathbf{dist}_{G[S]}(t, x) + \omega(\rho) \leq D/2$.

Next, it suffices to prove that $\rho \cap \mathbf{B}^-_{G[S]}(t, D/2) = \emptyset$. Suppose otherwise, then there exists $z \in \rho \cap \mathbf{B}^-_{G[S]}(t, D/2)$. Hence, $\mathbf{dist}_{G[S]}(x, t) \leq \omega(\rho) + \mathbf{dist}_{G[S]}(z, t) < 3D/4$, contradicting the fact that $x \in C_1$. ∎

A recursive application of (the contrapositive of) the above lemma immediately implies the soundness of our algorithm.

Corollary 3.1: Suppose $\mathrm{MinDiameter}(V, V)$ successfully returns a pair (t, z), then $\mathbf{dist}^{\min}_G(t, z) \geq D/4$.

The next lemma proves completeness of the algorithm.

Lemma 3.3: Consider a call to Algorithm $\mathsf{MinDiameter}(V, V)$. If there are two vertices $u, v \in V$ and $\mathbf{dist}^{\min}_G(u, v) \geq D$, then the invocation will return a pair (t, z) such that $\mathbf{dist}^{\min}_G(t, z) \geq D/4$.

Proof: Since $t \in C \setminus (C_1 \cup C_2)$, we have $|C_1|, |C_2| < |C|$, so the algorithm always terminates. Next, it suffices to prove that u, v are contained in the same set C on a root-to-leaf path on the recursion tree. As the basis, it is clear that $u, v \in V$.

Consider any recursion node (S, C) such that $u, v \in C$. If the algorithm terminates on line-8, then by definition of U_1, U_2, we have $\mathbf{dist}^{\min}_{G[S]}(t, z) \geq D/4$. Using Corollary 3.1, we can conclude the proof.

Next, assume the algorithm reaches line-9. By Lemma 3.1, either $u, v \in C_1$ or $u, v \in C_2$. So we can choose the branch $(S_b, C_b), b \in \{1, 2\}$ such that $C_b \ni u, v$ and continue with our induction. ∎

B. Running time analysis

Clearly, each recursion of the algorithm takes $\tilde{O}(|G[S]|)$ time (for invoking twice Dijkstra's algorithm on $G[S]$). By disjointness S_1, S_2, to bound the running time by near-linear, it suffices to bound the recursion depth.

Lemma 3.4: W.h.p. the recursion depth of Algorithm $\mathsf{MinDiameter}(V, V)$ is $O(\log n)$.

Proof: It is sufficient to show that at each recursion node (S, C) that reaches line-9, with at least constant probability, $|C_1|, |C_2| \leq 8|C|/9$.

In fact, noticing that for any $u \in C_1$, $\mathbf{dist}_{G[S]}(t, u) < D/4 < 3D/4 \leq \mathbf{dist}_{G[S]}(u, t)$, we know that $C_1 \subseteq X_{G[S]}(t)$; similarly we have $C_2 \subseteq Y_{G[S]}(t)$. Thus, applying Lemma 2.1 (with $U \leftarrow C$) with probability at least $1/2$ we have

$$|X_{G[S]}(t) \cap C|/8 \leq |Y_{G[S]}(t) \cap C| \leq 8|X_{G[S]}(t) \cap C|$$

Hence, $|C_1|, |C_2| \leq 8|C|/9$. ■

IV. Min-radius and min-eccentricities in general graphs

To prove Theorem 1.2, we will first describe the algorithm for min-radius, and then extend it for min-eccentricities. For min-radius, it suffices to design a near-linear time algorithm that, given any threshold R, either decides that the min-radius of G is larger than R, or finds a center vertex $c \in V$ such that $\max_{v \in V} \mathbf{dist}^{\min}(c, v) \leq 4R$. Since the input graph G has positive integral edge weights which are polynomially bounded, we can perform a binary search on R which gives a 4-approximation of min-radius of G.

The key component of our main algorithm is a recursive subroutine MinRadius that accepts three parameters $S, T, C \subseteq V$ which are vertex subsets satisfying $T \cup C \subseteq S$, and it either asserts that C does not contain any R-min-center, or finds a vertex $c \in C$ such that $\max_{u \in T} \mathbf{dist}^{\min}_{G[S]}(c, u) \leq 4R$, where $G[S]$ is the induced subgraph on vertex set S. The main algorithm simply invokes this subroutine MinRadius(V, V, V) to look for $4R$-min-centers in G.

Notations. Each node of the recursion tree $\mathcal{T}$ is specified by a triple of vertex set (S, T, C) which are input to the recursive subroutine MinRadius; so at the root node of $\mathcal{T}$, the triple is (V, V, V). In the end, we will ensure that the recursion tree $\mathcal{T}$ of this subroutine is always a binary tree whose depth is bounded by $O(\log n)$.

During the algorithm, we will explicitly store the entire configuration of the recursion tree $\mathcal{T}$. Also, for each vertex $u \in V$ and each tree node $N = (S, T, C)$ such that $u \in S$, we store an auxiliary label $L_u^N \in \{-1, 1, \perp\}$, where $\perp$ means "undefined".

Recursion. Next let us describe how the recursion subroutine works. If $C = \emptyset$, then our algorithm simply returns null. If $|C| \leq 10 \log n$, then for each $c \in C$ apply Dijkstra's algorithm to compute single-source shortest paths to and from c in subgraph $G[S]$, and check if there exists a vertex $c \in C$ such that $\max_{u \in T} \mathbf{dist}^{\min}_{G[S]}(c, u) \leq R$.

The nontrivial case is when $|C| > 10 \log n$. In this case, take a uniformly random vertex $t \in C$, and compute single-source shortest paths to and from t in the induced subgraph $G[S]$ by Dijkstra's algorithm. Then, compute the set $W = S \setminus (\mathbf{B}^+_{G[S]}(t, 2R) \cup \mathbf{B}^-_{G[S]}(t, 2R))$. Next, consider two different cases.

- **Leaf.** $T \subseteq \mathbf{B}^+_{G[S]}(t, 4R) \cup \mathbf{B}^-_{G[S]}(t, 4R)$.
  In this case, return t as a $4R$-min-center of the entire graph G.
- **Branch or prune.** T is not contained in $\mathbf{B}^+_{G[S]}(t, 4R) \cup \mathbf{B}^-_{G[S]}(t, 4R)$.
  In this case, the recursive algorithm will grow two branches. Let N be the current node (S, C, T) on the recursion tree $\mathcal{T}$. The first branch is defined as follows. Define a set
  $$S_1 = S \setminus (\mathbf{B}^-_{G[S]}(t, 2R) \cup L_1)$$
  where $L_1 \subseteq W$ is the set of all vertices $v \in W$ such that either $L_v^N = -1$ or there exists an ancestor node N' of N satisfying $L_v^{N'} = -1$.
  If $L_1 \cap T \neq \emptyset$, then we prune the first branch; namely the algorithm will not recurse further on this branch. Otherwise, define
  $$T_1 = S_1 \cap T \setminus \mathbf{B}^-_{G[S]}(t, 3R)$$
  $$C_1 = C \cap S_1 \cap \mathbf{B}^+_{G[S]}(t, R) \setminus \mathbf{B}^-_{G[S]}(t, 3R)$$
  and define a new node $N_1 = (S_1, T_1, C_1)$ on $\mathcal{T}$. Then for each $v \in S_1 \cap W$, set a label $L_v^{N_1} = 1$. After that, recurse on node N_1.
  The second branch is defined in the symmetric manner. Namely, take
  $$S_2 = S \setminus (\mathbf{B}^+_{G[S]}(t, 2R) \cup L_2)$$
  where $L_2 \subseteq W$ is the set of all vertices $v \in W$ such that either $L_v^N = 1$, or there exists an ancestor node N' of N satisfying $L_v^{N'} = 1$.
  If $L_2 \cap T \neq \emptyset$, then we prune the second branch. Otherwise, define
  $$T_2 = S_2 \cap T \setminus \mathbf{B}^+_{G[S]}(t, 3R)$$
  $$C_2 = C \cap S_2 \cap \mathbf{B}^-_{G[S]}(t, R) \setminus \mathbf{B}^+_{G[S]}(t, 3R)$$
  and define a new node $N_2 = (S_2, T_2, C_2)$. For each $v \in S_2 \cap W$, define a label $L_v^{N_2} = -1$. Finally, recurse on node N_2.

The recursion subroutine MinRadius is summarized as Algorithm 2.

A. Proof of correctness

Here is a basic property of our recursive algorithm, which is clearly guaranteed by the algorithm description.

Lemma 4.1: Let $N = (S, T, C)$ and $N' = (S', T', C')$ be two nodes on the recursion tree $\mathcal{T}$ such that N' is an ancestor of N. Then $S \subseteq S', T \subseteq T', C \subseteq C'$.

Next we argue that the set C provides good candidates for min-centers.

Lemma 4.2: Consider any node $N = (S, T, C)$ on $\mathcal{T}$. Then, for any pairs of vertices $c \in C, v \in V \setminus T$, $\mathbf{dist}^{\min}_G(c, v) \leq 4R$.

Proof: The statement is proved by an induction on the depth of nodes on the recursion tree. As the basis, when $N = (V, V, V)$ is the root node, clearly it satisfies the statement as $V \setminus T = \emptyset$. Next, let us focus on the inductive step.

Algorithm 2: MinRadius(S, T, C)

1 **if** $C = \emptyset$ **then**
2 **return** null;
3 **else if** $0 < |C| \leq 10 \log n$ **then**
4 for each $c \in C$ apply Dijkstra's algorithm to compute single-source shortest paths in $G[S]$;
5 **return** an arbitrary vertex $c \in C$ such that $\max_{u \in T} \mathbf{dist}^{\min}_{G[S]}(c, u) \leq R$;
6 **else**
7 take a random vertex $t \in C$, and compute single-source shortest paths at t in $G[S]$;
8 define $W = S \setminus (\mathbf{B}^{+}_{G[S]}(t, 2R) \cup \mathbf{B}^{-}_{G[S]}(t, 2R))$;
9 **if** $T \subseteq \mathbf{B}^{+}_{G[S]}(t, 4R) \cup \mathbf{B}^{-}_{G[S]}(t, 4R)$ **then**
10 **return** t;
11 **else**
  /* grow the first branch */
12 compute $L_1 \subseteq W$ which is set of all vertices $v \in W$ such that either $L^N_v = -1$ or there exists an ancestor node N' of N satisfying $L^{N'}_v = -1$;
13 $S_1 \leftarrow S \setminus (\mathbf{B}^{-}_{G[S]}(t, 2R) \cup L_1)$, $T_1 \leftarrow S_1 \cap T \setminus \mathbf{B}^{-}_{G[S]}(t, 3R)$, $C_1 \leftarrow C \cap S_1 \cap \mathbf{B}^{+}_{G[S]}(t, R) \setminus \mathbf{B}^{-}_{G[S]}(t, 3R)$;
14 set labels $L^{N_1}_v \leftarrow 1$ for each $v \in S_1 \cap W$;
15 create new child nodes $N_1 = (S_1, T_1, C_1)$ on the recursion tree $\mathcal{T}$;
16 **if** $L_1 \cap T = \emptyset$ **then**
17 run $c_1 \leftarrow$ MinRadius(S_1, T_1, C_1);
  /* grow the second branch */
18 define $N_2 = (S_2, C_2, T_2)$ in the symmetric manner;
19 set labels $L^{N_2}_v \leftarrow -1$ for each $v \in S_2 \cap W$;
20 **if** $L_2 \cap T = \emptyset$ **then**
21 run $c_2 \leftarrow$ MinRadius(S_2, T_2, C_2);
22 if c_1 is not null, then **return** c_1; else **return** c_2;

Suppose the statement holds for some node $N = (S, T, C)$. We will prove that in the case where the first branch is not pruned, node $N_1 = (S_1, T_1, C_1)$ will also satisfy the statement; we can repeat the symmetric argument for N_2 as well.

By Lemma 4.1, $T_1 \subseteq T, C_1 \subseteq C$, so it suffices to prove that for any vertex $c \in C_1$ and any vertex $v \in T \setminus T_1$, we have $\mathbf{dist}^{\min}(c, v) \leq 4R$. In fact, as $v \in T \setminus T_1$ and by definitions $S_1 = S \setminus (\mathbf{B}^{-}_{G[S]}(t, 2R) \cup L_1)$ and $T_1 = S_1 \cap T \setminus \mathbf{B}^{-}_{G[S]}(t, 3R)$, we know $T \setminus T_1 \subseteq \mathbf{B}^{-}_{G[S]}(t, 3R) \cup L_1$. Since N_1 is not pruned, $L_1 \cap T = \emptyset$. Therefore it must be $v \in \mathbf{B}^{-}_{G[S]}(t, 3R)$. So, by triangle inequality, we have $\mathbf{dist}^{\min}_{G}(v, c) \leq \mathbf{dist}_{G[S]}(v, c) \leq \mathbf{dist}_{G[S]}(v, t) + \mathbf{dist}_{G[S]}(t, c) \leq 4R$. ∎

As a corollary, the algorithm always returns a good candidate center.

Corollary 4.1: If the algorithm MinRadius(V, V, V) successfully returns any vertex c as a candidate min-center, then its min-eccentricity is at most $4R$.

Proof: If a candidate min-center c is returned on line-5 or line-10, then by the algorithm description, for any $u \in T$, $\mathbf{dist}^{\min}(c, u) \leq \mathbf{dist}^{\min}_{G[S]}(c, u) \leq 4R$. By Lemma 4.2, for any $v \in V \setminus T$, we also have $\mathbf{dist}^{\min}(c, v) \leq 4R$. Therefore, the min-eccentricity of c is at most $4R$. ∎

The following lemma explains the motivation of using labels L^N_v.

Lemma 4.3: If $L^{N_1}_v = 1$ for some node $N_1 = (S_1, T_1, C_1)$, then for any $c \in C_1$, $\mathbf{dist}_{G[S_1]}(c, v) > R$. Similarly, if $L^{N_2}_v = -1$ for some node $N_2 = (S_2, T_2, C_2)$, then for any $c \in C_2$, $\mathbf{dist}_{G[S_2]}(v, c) > R$.

Proof: Let us prove the statement for N_1 and a symmetric argument holds for N_2. By the algorithm description, it never assigns labels for the root node of $\mathcal{T}$, therefore N_1 is not the root node. Suppose $N = (S, T, C)$ is the parent node of N_1. As $L^{N_1}_v = 1$, we can assume N_1 is on the first branch below N. Then $c \in \mathbf{B}^{+}_{G[S]}(t, R)$. By the way we set the labels, we know $v \in W$. By definition of W, $\mathbf{dist}_{G[S]}(t, v) > 2R$, and so using triangle inequality we have:

$$\begin{aligned}\mathbf{dist}_{G[S_1]}(c, v) &\geq \mathbf{dist}_{G[S]}(c, v) \\ &\geq \mathbf{dist}_{G[S]}(t, v) - \mathbf{dist}_{G[S]}(t, c) > R\end{aligned}$$

∎

Assume G has min-radius at most R. Then there exists a center c^* whose eccentricity in G is at most R. In the next lemma, we prove that c^* always lies in some set C.

Lemma 4.4: Vertex c^* is contained in the set C for all nodes $N = (S, T, C)$ lying on a certain root-to-leaf tree path of $\mathcal{T}$. Plus, for any such node N, for every vertex $c \in C$ and vertex $v \in T \cup C$, if $\mathbf{dist}_G(c, v) \leq R$, then $\mathbf{dist}_G(c, v) = \mathbf{dist}_{G[S]}(c, v)$; symmetrically, if $\mathbf{dist}_G(v, c) \leq R$, then $\mathbf{dist}_G(v, c) = \mathbf{dist}_{G[S]}(v, c)$.

Proof: The statement is proved by an induction on recursion depth. As the basis, the statement holds trivially for the root node. Now suppose at a certain depth, there is a node $N = (S, T, C)$ that satisfies both properties. Suppose it enters the branch-or-prune phase and reaches line-12 of Algorithm 2; otherwise this node is already a leaf node.

By the inductive hypothesis, as $t \in C$, we know $\mathbf{dist}^{\min}_{G[S]}(c^*, t) \leq R$. Therefore, $c^* \in \mathbf{B}^{+}_{G[S]}(t, R) \cup \mathbf{B}^{-}_{G[S]}(t, R)$. Without loss of generality, assume $c^* \in \mathbf{B}^{+}_{G[S]}(t, R)$.

Claim 4.4.1: $c^* \notin \mathbf{B}^{-}_{G[S]}(t, 3R)$.

Proof of claim: By the branch-or-prune condition, we know T is not contained in $\mathbf{B}^{+}_{G[S]}(t, 4R) \cup \mathbf{B}^{-}_{G[S]}(t, 4R)$. So there exists $v \in T$ such that $\mathbf{dist}^{\min}_{G[S]}(t, v) > 4R$. By triangle inequality,

$$\begin{aligned}\mathbf{dist}_{G[S]}(c^*, v) &\geq \mathbf{dist}_{G[S]}(t, v) - \mathbf{dist}_{G[S]}(t, c^*) \\ &> 4R - R = 3R > R\end{aligned}$$

Then, by the contrapositive of the inductive hypothesis, we know $\mathbf{dist}_G(c^*, v) > R$. As c^* is an R-min-center, we know

$\mathbf{dist}_G(v, c^*) \leq R$. So again by the inductive hypothesis and triangle inequality,

$$\begin{aligned} 4R - \mathbf{dist}_{G[S]}(c^*, t) &< \mathbf{dist}_{G[S]}(v, t) - \mathbf{dist}_{G[S]}(c^*, t) \\ &\leq \mathbf{dist}_{G[S]}(v, c^*) \leq R \end{aligned}$$

So $\mathbf{dist}_{G[S]}(c^*, t) > 3R$, or equivalently $c^* \notin \mathbf{B}^-_{G[S]}(t, 3R)$. ∎

By the above claim and definition of C_1 and the fact that $L_1 \subseteq W$ we know $c^* \in C_1$. Next we need to verify that the branch on N_1 is not pruned.

Claim 4.4.2: $L_1 \cap T = \emptyset$.

Proof of claim: Suppose otherwise there exists a vertex $v \in L_1 \cap T$. Then by definition of L_1, either $L_v^N = -1$, or there exists an ancestor $N' = (S', T', C')$ such that $L_v^{N'} = -1$. By Lemma 4.1 and Lemma 4.3, we have $\mathbf{dist}_{G[S]}(v, c^*) > R$. Using the inductive hypothesis on $v \in T$ and $c^* \in C$, we know $\mathbf{dist}_G(v, c^*) > R$ as well.

Now, as c^* is an R-min-center, it must be $\mathbf{dist}_G(c^*, v) \leq R$. So, by inductive hypothesis, $\mathbf{dist}_{G[S]}(c^*, v) = \mathbf{dist}_G(c^*, v) \leq R$. By triangle inequality, $\mathbf{dist}_{G[S]}(t, v) \leq \mathbf{dist}_{G[S]}(t, c^*) + \mathbf{dist}_{G[S]}(c^*, v) \leq 2R$, which contradicts the fact that $v \in L_1 \subseteq W$. ∎

Now let us turn to the second half of the statement. Consider any vertex $v \in T_1 \cup C_1$ and $c \in C_1$ such that $\mathbf{dist}^{\min}_G(c, v) \leq R$. By the inductive hypothesis, $\mathbf{dist}^{\min}_{G[S]}(c, v) \leq R$. Consider two possibilities.

- $\mathbf{dist}_{G[S]}(c, v) \leq R$.
  In this case, let ρ be the shortest path from c to v in $G[S]$. We show that ρ lies entirely in S_1, or equivalently, that ρ does not intersect $\mathbf{B}^-_{G[S]}(t, 2R) \cup L_1$, which would prove $\mathbf{dist}_{G[S_1]}(c, v) = \mathbf{dist}_{G[S]}(c, v) = \mathbf{dist}_G(c, v)$.
  If there is a vertex $z \in \rho \cap \mathbf{B}^-_{G[S]}(t, 2R)$, then by triangle inequality, $\mathbf{dist}_{G[S]}(c, t) \leq \omega(\rho) + \mathbf{dist}_{G[S]}(z, t) \leq 3R$, which contradicts $c \notin \mathbf{B}^-_{G[S]}(t, 3R)$ as $C_1 \cap \mathbf{B}^-_{G[S]}(t, 3R) = \emptyset$.
  If there is a vertex $z \in \rho \cap L_1$, then $\mathbf{dist}_{G[S]}(c, z) \leq \omega(\rho) \leq R$. So by triangle inequality, $\mathbf{dist}_{G[S]}(t, z) \leq \mathbf{dist}_{G[S]}(t, c) + \mathbf{dist}_{G[S]}(c, z) \leq 2R$, which contradicts $z \in L_1 \subseteq W$.
- $\mathbf{dist}_{G[S]}(v, c) \leq R$.
  In this case, let ρ be the shortest path from v to c in $G[S]$. We also show that ρ lies entirely in S_1, or equivalently, that ρ does not intersect $\mathbf{B}^-_{G[S]}(t, 2R) \cup L_1$, which would prove $\mathbf{dist}_{G[S_1]}(v, c) = \mathbf{dist}_{G[S]}(v, c) = \mathbf{dist}_G(v, c)$.
  If there is a vertex $z \in \rho \cap \mathbf{B}^-_{G[S]}(t, 2R)$, then by triangle inequality, $\mathbf{dist}_{G[S]}(v, t) \leq \omega(\rho) + \mathbf{dist}_{G[S]}(z, t) \leq 3R$, which contradicts $v \notin \mathbf{B}^-_{G[S]}(t, 3R)$ as $(T_1 \cup C_1) \cap \mathbf{B}^-_{G[S]}(t, 3R) = \emptyset$.
  If there is a vertex $z \in \rho \cap L_1$, then $\mathbf{dist}_{G[S]}(z, c) \leq \omega(\rho) \leq R$. However, by definition of L_1, either $L_z^N = -1$, or there exists an ancestor $N' = (S', T', C')$ of N such that $L_z^{N'} = -1$. By Lemma 4.3, $\mathbf{dist}_{G[S]}(z, c) > R$, contradiction. ∎

B. Running time analysis

First we analyze the depth of the recursion tree.

Lemma 4.5: With high probability, the depth of the recursion tree $\mathcal{T}$ is at most $O(\log n)$.

Proof: Consider any branch-or-prune phase at any node $N = (S, T, C)$. It suffices to prove that with constant probability over the choice of $t \in C$, we have $\max\{|C_1|, |C_2|\} \leq \frac{8}{9}|C|$. By Lemma 2.1, with probability at least $1/2$ over the choice of $t \in C$, we have

$$|X_{G[S]}(t) \cap C|/8 \leq |Y_{G[S]}(t) \cap C| \leq 8|X_{G[S]}(t) \cap C|$$

Now, by construction of C_1, for any $c \in C_1$, $\mathbf{dist}_{G[S]}(t, c) \leq R < 3R < \mathbf{dist}_{G[S]}(c, t)$, and so $c \in X_{G[S]}(t)$. Therefore, $|C_1| \leq \frac{8}{9}|C|$. Similarly we also have $|C_2| \leq \frac{8}{9}|C|$. ∎

Secondly, we study the total cost of recursion. The following lemma explains the motivation of using labels.

Lemma 4.6: For any vertex $v \in V$, v belongs to S for at most $O(\log n)$ different nodes $N = (S, T, C)$ of the recursion tree $\mathcal{T}$.

Proof: Consider the branch-or-prune phase of any node $N = (S, T, C)$ such that $S \ni v$, and let $N_1 = (S_1, T_1, C_1), N_2 = (S_2, T_2, C_2)$ be its two children. If vertex v is always passed to at most one branch, namely $v \notin S_1$ or $v \notin S_2$, then by Lemma 4.5, as the tree depth is bounded by $O(\log n)$ with high probability, v belongs to set S of at most $O(\log n)$ different nodes.

Now, consider the first time that v belongs to both S_1 and S_2. Then, by the algorithm description, v must belong to W, and so it must have set the labels $L_v^{N_1} = 1$ and $L_v^{N_2} = -1$. We claim that, for any descendant node $M = (S_0, T_0, C_0)$ of $N_b, \forall b \in \{1, 2\}$ on the recursion tree $\mathcal{T}$ such that $v \in S_0$, v is passed on to at most one child of M during the branch-or-prune procedures at M. Say, M is a descendant of N_1. During the branch-or-prune phase at M, let $W_0 = S_0 \setminus (\mathbf{B}^+_{G[S_0]}(t, 2R) \cup \mathbf{B}^-_{G[S_0]}(t, 2R))$, where t is a uniformly random vertex taken from C_0, and let M_1, M_2 be the two children of M. If $v \notin W_0$, then by the algorithm v is passed to at most one of M_1, M_2 in deeper recursions; otherwise if $v \in W_0$, then since $L_v^{N_1} = 1$ and N_1 is an ancestor of M, v can never be passed on to the second branch M_2. A symmetric argument works with N_2.

By the above claim, all descendants of N_b that include v should form an ancestor-to-descendant tree path starting at $N_b, \forall b \in \{1, 2\}$. Therefore, as the depth of $\mathcal{T}$ is bounded by $O(\log n)$, v belongs to at most $O(\log n)$ nodes. See Figure 1 for an illustration. ∎

By the above lemma, each vertex is copied for at most $O(\log n)$ times, so the total cost of running Dijkstra's algorithm is bounded by $\tilde{O}(m)$.

C. Extension to eccentricities

Our approximate min-eccentricity algorithm follows the basic framework of the previous min-radius algorithm, with slight modifications. Take $R = 1, \lceil(1 + \epsilon/5)\rceil, \lceil(1 + \epsilon/5)^2\rceil, \cdots, \lceil(1 + \epsilon/5)^i\rceil, \cdots$, and for each R we will

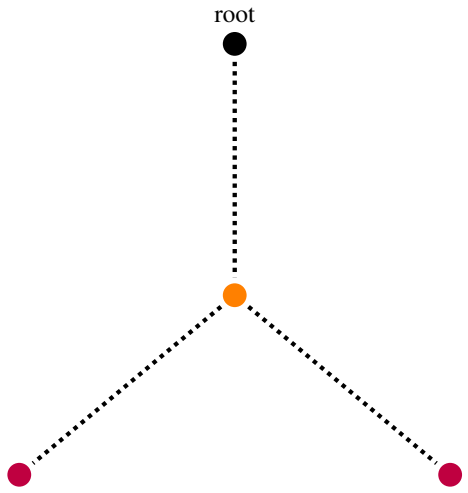

Fig. 1. In this picture, the black node is the root of $\mathcal{T}$. The orange node is the highest node where v belongs to both S_1, S_2. Then, all descendants of the orange node that involves v should form two tree paths. Overall, all nodes that contain v form two tree paths starting at the root, which are shown as the dotted lines. As the depth of $\mathcal{T}$ is $O(\log n)$, the length of the dotted lines is also $O(\log n)$.

give a near-linear time algorithm that for each $u \in V$, either decides $\max_{v\in V} \mathbf{dist}^{\min}(u,v) > R$ or certifies $\max_{v\in V} \mathbf{dist}^{\min}(u,v) \le 5R$; if this can be done, then we have a near-linear time algorithm that approximates all min-eccentricities within stretch $5+\epsilon$. The full algorithm is summarized as Algorithm 3. At the root of the recursion, we apply $\mathsf{MinEcc}(V,V,V)$. For general inputs (S,T,C), the algorithm decides for each $c \in C$ whether the eccentricity of c is larger than R or at most $5R$ in G.

The algorithm is mostly the same as Algorithm 2. So here we only discuss the modifications in the recursion step.

Recursion. Let $N = (S,T,C)$ be the current recursion node. In the case where $|C| > 10\log n$, take a uniformly random vertex $t \in C$ and compute single-source shortest paths to and from t in subgraph $G[S]$ by Dijkstra's algorithm. Define $W = S \setminus (\mathbf{B}^+_{G[S]}(t,2R) \cup \mathbf{B}^-_{G[S]}(t,2R))$.

If $T \subseteq \mathbf{B}^+_{G[S]}(t,4R) \cup \mathbf{B}^-_{G[S]}(t,4R)$, then identify t as a $5R$-min-center of G. Next we need to conduct further recursions to identify the rest of C. Similar to Algorithm 2, the algorithm grows two branches; the two branches are symmetric so we only define the first branch.

Define $L_1 \subseteq W$ is the set of all vertices $v \in W$ such that either $L_v^N = -1$, or there exists an ancestor node N' of N with $L_v^{N'} = -1$. Then, take $S_1 = S \setminus (\mathbf{B}^-_{G[S]}(t,2R) \cup L_1)$. If $L_1 \cap T \neq \emptyset$, then the first branch is pruned. Otherwise, define

$$T_1 = S_1 \cap T \setminus \mathbf{B}^-_{G[S]}(t,3R)$$
$$C_1 = C \cap S_1 \cap \mathbf{B}^+_{G[S]}(t,R) \setminus \mathbf{B}^-_{G[S]}(t,3R)$$

and define a new node $N_1 = (S_1, T_1, C_1)$. Then for each $v \in S_1 \cap W$, define a label $L_v^{N_1} = 1$.

Before recurring on node N_1, we need to identify vertices in

$$C \cap \mathbf{B}^+_{G[S]}(t,R) \setminus C_1 = C \cap \mathbf{B}^+_{G[S]}(t,R) \cap \mathbf{B}^-_{G[S]}(t,3R)$$

as either $5R$-min-centers, or assert their eccentricities are larger than R, which is the main modification to Algorithm 2. If $T \subseteq \mathbf{B}^+_{G[S]}(t,2R) \cup \mathbf{B}^-_{G[S]}(t,4R)$, then the algorithm reports all vertices in $C \cap \mathbf{B}^+_{G[S]}(t,R) \cap \mathbf{B}^-_{G[S]}(t,3R)$ as $5R$-min-centers; otherwise, the algorithm claims that all vertices in $C \cap \mathbf{B}^+_{G[S]}(t,R) \cap \mathbf{B}^-_{G[S]}(t,3R)$ have eccentricities larger than R.

For the runtime analysis, the only extra part of Algorithm 3 is checking if $T \subseteq \mathbf{B}^+_{G[S]}(t,2R) \cup \mathbf{B}^-_{G[S]}(t,4R), T \subseteq \mathbf{B}^-_{G[S]}(t,2R) \cup \mathbf{B}^+_{G[S]}(t,4R)$ and identifying $5R$-min-centers in $C \cap \mathbf{B}^+_{G[S]}(t,R) \cap \mathbf{B}^-_{G[S]}(t,3R)$ and $C \cap \mathbf{B}^-_{G[S]}(t,R) \cap \mathbf{B}^+_{G[S]}(t,3R)$. This takes time linear in the size of S, so it is does not change the asymptotic running time.

Following similar arguments in Lemma 4.4, we can prove that short distances between C and $T \cup C$ are preserved in $G[S]$. It suffices to analyze the min-eccentricity of vertices in $C \cap \mathbf{B}^+_{G[S]}(t,R) \cap \mathbf{B}^-_{G[S]}(t,3R)$ and $C \cap \mathbf{B}^-_{G[S]}(t,R) \cap \mathbf{B}^+_{G[S]}(t,3R)$.

Lemma 4.7: During the recursion, for any vertex $c \in C \cap \mathbf{B}^+_{G[S]}(t,R) \cap \mathbf{B}^-_{G[S]}(t,3R)$, if $T \subseteq \mathbf{B}^+_{G[S]}(t,2R) \cup \mathbf{B}^-_{G[S]}(t,4R)$, then c is a $5R$-min-center of G; otherwise, the min-eccentricity of c in G is larger than R. Symmetrically, the statement holds for any $c \in C \cap \mathbf{B}^-_{G[S]}(t,R) \cap \mathbf{B}^+_{G[S]}(t,3R)$ as well.

Proof: Suppose $T \subseteq \mathbf{B}^+_{G[S]}(t,2R) \cup \mathbf{B}^-_{G[S]}(t,4R)$. Consider any $c \in C \cap \mathbf{B}^+_{G[S]}(t,R) \cap \mathbf{B}^-_{G[S]}(t,3R)$ and $v \in T$. If $v \in \mathbf{B}^+_{G[S]}(t,2R)$, then $\mathbf{dist}_{G[S]}(c,v) \le \mathbf{dist}_{G[S]}(c,t) + \mathbf{dist}_{G[S]}(t,v) \le 3R + 2R = 5R$; if $v \in \mathbf{B}^-_{G[S]}(t,4R)$, then $\mathbf{dist}_{G[S]}(v,c) \le \mathbf{dist}_{G[S]}(v,t) + \mathbf{dist}_{G[S]}(t,c) \le 4R + R = 5R$. By Lemma 4.2, the min-distance between c and vertices not in T is bounded by $4R$, so c is a $5R$-min-center of G.

Now, suppose T is not in $\mathbf{B}^+_{G[S]}(t,2R) \cup \mathbf{B}^-_{G[S]}(t,4R)$, then there exists $z \in T \setminus (\mathbf{B}^+_{G[S]}(t,2R) \cup \mathbf{B}^-_{G[S]}(t,4R))$. By triangle inequality, $\mathbf{dist}_{G[S]}(z,c) > 4R - 3R = R$, $\mathbf{dist}_{G[S]}(c,z) > 2R - R = R$. By Lemma 4.4, the min-distance between c, z in G is larger than R. ∎

V. Min-radius and min-eccentricities in acyclic graphs

In this section we present our algorithm underlying Theorem 1.3. Let $G = (V,E,\omega)$ be an acyclic directed graph with positive integral edge weights. Let $\pi : V \to [n]$ be a topological ordering of vertices in V, so that for any $u, v \in V$, if u can reach v then $\pi(u) < \pi(v)$. To unify the algorithms for radius and eccentricities, it suffices to prove the following lemma.

Lemma 5.1: For any positive integer R, there is a near-linear time randomized algorithm that computes a vertex subset $C^* \subseteq V$, such that for any $c \in C^*$, the min-eccentricity of c

Algorithm 3: $\mathsf{MinEcc}(S, T, C)$

if $0 < |C| \leq 10 \log n$ **then**
  for each $c \in C$ apply Dijkstra's algorithm to compute single-source shortest paths in $G[S]$;
  for each $c \in C$ such that $\max_{v \in T} \mathbf{dist}^{\min}_{G[S]}(c, v) \leq R$, identify c as a $5R$-min-center in G;
else
  take a random vertex $t \in C$, and compute single-source shortest paths at t in $G[S]$;
  define $W = S \setminus (\mathbf{B}^+_{G[S]}(t, 2R) \cup \mathbf{B}^-_{G[S]}(t, 2R))$;
  if $T \subseteq \mathbf{B}^+_{G[S]}(t, 4R) \cup \mathbf{B}^-_{G[S]}(t, 4R)$ **then**
    identify t as a $5R$-min-center;
  `/* grow the first branch */`
  compute $L_1 \subseteq W$ which is set of all vertices $v \in W$ such that either $L^N_v = -1$ or there exists an ancestor node N' of N satisfying $L^{N'}_v = -1$;
  $S_1 \leftarrow S \setminus (\mathbf{B}^-_{G[S]}(t, 2R) \cup L_1)$,
  $T_1 \leftarrow S_1 \cap T \setminus \mathbf{B}^-_{G[S]}(t, 3R)$,
  $C_1 \leftarrow C \cap S_1 \cap \mathbf{B}^+_{G[S]}(t, R) \setminus \mathbf{B}^-_{G[S]}(t, 3R)$;
  set labels $L^{N_1}_v \leftarrow 1$ for each $v \in T_1 \cap W$;
  create new child nodes $N_1 = (S_1, T_1, C_1)$ on the recursion tree $\mathcal{T}$;
  `/* main modifications here */`
  if $T \subseteq \mathbf{B}^+_{G[S]}(t, 2R) \cup \mathbf{B}^-_{G[S]}(t, 4R)$ **then**
    identify all vertices in $C \cap \mathbf{B}^+_{G[S]}(t, R) \cap \mathbf{B}^-_{G[S]}(t, 3R)$ as $5R$-min-centers;
  if $T \subseteq \mathbf{B}^-_{G[S]}(t, 2R) \cup \mathbf{B}^+_{G[S]}(t, 4R)$ **then**
    identify all vertices in $C \cap \mathbf{B}^-_{G[S]}(t, R) \cap \mathbf{B}^+_{G[S]}(t, 3R)$ as $5R$-min-centers;
  `/* grow the first branch */`
  if $L_1 \cap T = \emptyset$ **then**
    run $\mathsf{MinEcc}(S_1, T_1, C_1)$;
  `/* grow the second branch */`
  define $N_2 = (S_2, C_2, T_2)$ in the symmetric manner;
  set labels $L^{N_2}_v \leftarrow -1$ for each $v \in T_2 \cap W$;
  if $L_2 \cap T = \emptyset$ **then**
    run $\mathsf{MinEcc}(S_2, T_2, C_2)$;

is at most $3R$; plus that for any $v \notin C^*$, the min-eccentricity of c is larger than R.

If the above lemma is proved, then to approximate min-radius, we can conduct a binary search over all polynomially bounded integers R to find the smallest threshold such that a $3R$-min-center exists, which gives a 3-approximation. For approximate min-eccentricities, try difference choices of $R = 1, \lceil 1 + \epsilon/3 \rceil, \lceil (1 + \epsilon/3)^2 \rceil, \cdots, \lceil (1 + \epsilon/3)^{O(\log n/\epsilon)} \rceil$ and apply the above lemma, which gives $3 + \epsilon$-approximation of all eccentricities.

Notations. The algorithm underlying Lemma 5.1 is a recursive subroutine. Throughout the algorithm, starting with $C^* = \emptyset$, it keeps adding more and more vertices to C^*. Let $\mathcal{T}$ be the recursion tree. Each node of the recursion tree is associated with two parameters (C, P) below.

(i) $C \subseteq V$ is a set of candidate centers. Define vertex $c_{\text{med}} \in C$ such that $\pi(c_{\text{med}})$ takes the median in $\{\pi(v) \mid v \in C\}$. We will ensure that different tree nodes will have different values of c_{med}, so we can also use $c_{\text{med}} \in V$ to refer to nodes of $\mathcal{T}$.

(ii) $P \subseteq V$ refers to a subset of ancestors of c_{med} on $\mathcal{T}$.

At the root of the recursion tree, the algorithm takes parameter $C = V$ and $P = \emptyset$. Next we describe how the algorithm recurs.

Recursion. Consider any node (C, P) on the recursion tree $\mathcal{T}$. Suppose $C \neq \emptyset$. For each vertex $c \in V$ which is an ancestor of c_{med} in tree $\mathcal{T}$, if $\pi(c) < \pi(c_{\text{med}})$, then define $V_c = \{v \in V \mid \pi(v) \leq \pi(c)\}$; otherwise define $V_c = \{v \in V \mid \pi(v) \geq \pi(c)\}$. Define $S = V \setminus \bigcup_{c \in P} V_c$. Apply Dijkstra's algorithm in the induced subgraph $G[S]$ to compute single-source shortest paths to and from c_{med}. If all vertices in S are within min-distances at most $2R$ from c_{med}, then add c_{med} to C^*.

Next we try to create two child nodes of (C, P) which are specified as $(C_1, P_1), (C_2, P_2)$ for recursions. Let us only describe how to compute (C_1, P_1); (C_2, P_2) can be defined in a symmetric manner. Define $C_1 = C \cap \{c_1 \mid \mathbf{dist}_{G[S]}(c_{\text{med}}, c_1) \leq R\} \setminus \{c_{\text{med}}\}$. Go over every ancestor of c_{med} on $\mathcal{T}$ not in set P which can be specified by vertex $c \in V \setminus P$. To construct the set P_1, initialize $P_1 = P$, and then consider two cases below.

- Suppose $\pi(c) < \pi(c_{\text{med}})$.
  In this case, check if $\mathbf{B}^-_{G[S]}(c_{\text{med}}, 2R)$ covers all vertices in $V_c \cap S$. If so, then add c to P_1.
- Suppose $\pi(c) > \pi(c_{\text{med}})$.
  In this case, check if $\mathbf{B}^+_{G[S]}(\text{med}, 2R)$ covers all vertices in $V_c \cap S$. If not, then prune the node (C_1, P_1) entirely.

After that, if (C_1, P_1) is not pruned, then append it as a child below (C, P) on $\mathcal{T}$ and recurse further. This recursive subroutine is summarized as Algorithm 4.

A. Proof of correctness

Here is a basic property of the recursion tree $\mathcal{T}$.

Lemma 5.2: Consider any tree node (C, P) and any of its ancestor on $\mathcal{T}$ specified by vertex $c \in V$. Then, either

Algorithm 4: DagMinEcc(C, P)

1 **if** $C \neq \emptyset$ **then**
2 define $S = V \setminus \bigcup_{c \in P} V_c$;
3 apply Dijkstra's algorithm to compute single-source shortest paths to and from c_{med} in $G[S]$;
4 **if** $S = \mathbf{B}^+_{G[S]}(c_{\text{med}}, 2R) \cup \mathbf{B}^-_{G[S]}(c_{\text{med}}, 2R)$ **then**
5 $C^* \leftarrow C^* \cup \{c_{\text{med}}\}$;
6 compute sets $C_1 = C \cap \{c_1 \mid \mathbf{dist}_{G[S]}(c_{\text{med}}, c_1) \leq R\} \setminus \{c_{\text{med}}\}$ and $C_2 = C \cap \{c_2 \mid \mathbf{dist}_{G[S]}(c_2, c_{\text{med}}) \leq R\} \setminus \{c_{\text{med}}\}$, and initialize $P_1 = P_2 = P$;
7 **for** *ancestor* $c \in V \setminus P$ *of* c_{med} *in the recursion tree* $\mathcal{T}$ **do**
8 **if** $\pi(c) < \pi(c_{\text{med}})$ **then**
9 **if** $\mathbf{B}^-_{G[S]}(c_{\text{med}}, 2R)$ *covers all vertices in* $V_c \cap S$ **then**
10 $P_1 \leftarrow P_1 \cup \{c\}$;
11 **else**
12 prune the child (C_2, P_2) in later recursions;
13 **else**
14 **if** $\mathbf{B}^+_{G[S]}(c_{\text{med}}, 2R)$ *covers all vertices in* $V_c \cap S$ **then**
15 $P_2 \leftarrow P_2 \cup \{c\}$;
16 **else**
17 prune the child (C_1, P_1) in later recursions;
18 **if** (C_1, P_1) *was not pruned* **then**
19 then run DagMinEcc(C_1, P_1);
20 **if** (C_2, P_2) *was not pruned* **then**
21 then run DagMinEcc(C_2, P_2);

$\pi(c) < \min_{c' \in C} \pi(c')$, or $\pi(c) > \max_{c' \in C} \pi(c')$. Also, for any ancestor node (C', P') of (C, P), we have $C \subseteq C', P \subseteq P'$.

Proof: This is straightforward by the way we define sets C_1, C_2, P_1, P_2 during the recursion steps. ∎

Now we prove that C makes a good set of candidates for min-centers in some sense.

Lemma 5.3: Consider any tree node (C, P) of $\mathcal{T}$. Let sets V_c be defined as in the algorithm description, where c ranges over all ancestors of c_{med}. Then, for any $t \in C$ and $v \in \bigcup_{c \in P} V_c$, we have $\mathbf{dist}^{\min}_G(t, v) \leq 3R$.

Proof: This is proved by an induction on the depth of node (C, P). The statement holds trivially for the basis where $(C, P) = (V, \emptyset)$. Suppose the statement holds for node (C, P), then we will prove that its child nodes $(C_1, P_1), (C_2, P_2)$ also meet the requirement. Consider any vertex $c \in P_1 \setminus P$ and any vertex $v \in V_c \setminus \bigcup_{c' \in P} V_{c'} = V_c \cap S$. By the algorithm, since c is added to P_1, it must be the case that $\pi(c) < \pi(c_{\text{med}})$, and that $\mathbf{B}^-_{G[S]}(c_{\text{med}}, 2R)$ contains all vertices in $V_c \cap S$. Therefore, for any $v \in V_c \cap S$, $\mathbf{dist}_G(v, t) \leq \mathbf{dist}_{G[S]}(v, t) \leq \mathbf{dist}_{G[S]}(v, c_{\text{med}}) + \mathbf{dist}_{G[S]}(c_{\text{med}}, t) \leq 2R + R = 3R$. A symmetric argument also holds for (C_2, P_2), which completes the induction. ∎

Corollary 5.1: After DagMinEcc$(V, \emptyset)$ terminates, for any vertex $c \in C^*$, its min-eccentricity in G is at most $3R$.

Proof: Suppose the recursion added c_{med} to C^* on line-5 of Algorithm 4 with some pair of input (C, P). By the branching condition, all vertices in S are within min-distances of at most $2R$ from c_{med} in $G[S]$, thus in G as well. Also, by the above lemma, for each $v \in V \setminus S$, $\mathbf{dist}^{\min}_G(c_{\text{med}}, v) \leq 3R$. Hence, the eccentricity of c_{med} in G is at most $3R$. ∎

So far we have proved that all vertices in C^* are of min-eccentricities at most $3R$. Next we need to lower bound the min-eccentricities of vertices outside of C^*. Consider any vertex c^* whose min-eccentricity is at most R. We prove that c^* must have been added to C^* during the execution of DagMinEcc$(V, \emptyset)$.

Lemma 5.4: There is a tree path starting at the root on the recursion tree $\mathcal{T}$, such that for each node (C, P) on this path, $c^* \in C$, plus that c^* is added to C^* at the ending node of this tree path.

Proof: Let us prove the first half of the statement by an induction on depth. The basis trivially holds for the root node as $(C, P) = (V, \emptyset)$. For the inductive step, let us suppose that the statement holds for some pair (C, P). Consider three different cases.

(1) $c^* = c_{\text{med}}$.
In this case, the algorithm computes single-source shortest paths to and from c^* in $G[S]$ and checks if $S = \mathbf{B}^+_{G[S]}(c^*, 2R) \cup \mathbf{B}^-_{G[S]}(c^*, 2R)$. Noticing that by definition $S = V \setminus \bigcup_{c \in P} V_c$, each V_c is either $\{v \mid \pi(v) \leq \pi(c)\}$ or $\{v \mid \pi(v) \geq \pi(c)\}$, depending on the relative order between $\pi(c)$ and $\pi(c_{\text{med}})$, and so vertices from $S = V \setminus \bigcup_{c \in P} V_c$ are a consecutive interval in terms of order $\pi(\cdot)$. Therefore, for each $v \in S$, $\mathbf{dist}^{\min}_{G[S]}(c^*, v) = \mathbf{dist}^{\min}_G(c^*, v) \leq R$, and thus $S = \mathbf{B}^+_{G[S]}(c^*, 2R) \cup \mathbf{B}^-_{G[S]}(c^*, 2R)$. Hence, c^* is added to C^*.

(2) $\pi(c^*) > \pi(c_{\text{med}})$.
Clearly, by Lemma 5.2, we have $C \subseteq S$. As S forms a consecutive interval of V in terms of the topological ordering $\pi(\cdot)$, we have $\mathbf{dist}_{G[S]}(c_{\text{med}}, c^*) = \mathbf{dist}_G(c_{\text{med}}, c^*) \leq R$. So by the algorithm description, we have $c^* \in C_1$. To show c^* is inherited in the child node (C_1, P_1), we need to verify that (C_1, P_1) is never pruned by the algorithm. Suppose otherwise, then there exists an ancestor $c \in V \setminus P$, such that $\pi(c) > \pi(c_{\text{med}})$ and $V_c \cap S \setminus \mathbf{B}^+_{G[S]}(c_{\text{med}}, 2R)$ is nonempty, say $z \in V_c \cap S \setminus \mathbf{B}^+_{G[S]}(c_{\text{med}}, 2R)$. By Lemma 5.2, $\pi(z) \geq \pi(c) > \pi(c^*)$. As c^* is an R-min-center, $\mathbf{dist}_{G[S]}(c^*, z) = \mathbf{dist}_G(c^*, z) \leq R$. However, by triangle inequality, $\mathbf{dist}_{G[S]}(c^*, z) \geq \mathbf{dist}_{G[S]}(c_{\text{med}}, z) -$

$\mathbf{dist}_{G[S]}(c_{\text{med}}, c^*) > 2R - R = R$, which is a contraction.

(3) $\pi(c^*) < \pi(c_{\text{med}})$.
This case is symmetric to the previous one, so we neglect the analysis here.■

B. Running time analysis

It is easy to observe that the depth of $\mathcal{T}$ is always bounded by $O(\log n)$, as $|C_1|, |C_2| \leq |C|/2$ for any tree node (C, P). The running time is dominated by applying Dijkstra's algorithm to compute single-source shortest paths. By Lemma 5.2, it is straightforward to maintain the subset S along with P. So we only need to care about the total size of subgraphs $G[S]$ across all tree nodes. It suffices to prove the following lemma.

Lemma 5.5: For any vertex v, it belongs to set S for at most $O(\log^2 n)$ different tree nodes (C, P).

Proof: If we only look at the value of c_{med} of each node (C, P) of the recursion tree $\mathcal{T}$, then it corresponds to a binary search tree of all vertices from V. Fix any $c \in V$ which is an ancestor of v on $\mathcal{T}$; without loss of generality, assume $\pi(c) \geq \pi(v)$. Consider all the descendants t of c such that $\pi(t) > \pi(c)$.

We argue that v contributes to the SSSP instances of at most $O(\log n)$ of such descendants t of c during the recursion DagMinEcc. In fact, suppose the recursion at t has input pair (C, P) and t is the c_{med} vertex with respect to C. Then according to line-9 through line-12 of Algorithm 4, either c is added to P_1, or the child node (C_2, P_2) is pruned.

- If c is added to P_1, then according to Lemma 5.2, for any descendant of c specified by (C', P'), we have $c \in P_1 \subseteq P'$. As $\pi(v) \leq \pi(c) < \pi(t)$, v does not participate in the SSSP computation at node (C', P').
- If (C_2, P_2) is pruned, then v can only contribute to the first branch (C_1, P_1) of (C, P).

Therefore, v contributes to the SSSP instances of at most one of the two children of t. So, in the end, all tree nodes whose SSSP instances contain v should form a tree path starting from c to a leaf node. So the total contribution of v is bounded by $O(\log n)$. Summing over all different ancestors c, the overall contribution of c is bounded by $O(\log^2 n)$. See Figure 2 for an illustration. ■

By the above lemma, for each vertex $v \in V$, during all invocations of DagMinEcc, it appears in the SSSP instances for at most $O(\log^2 n)$ times. Therefore, the total cost of SSSP instances is bounded by $\tilde{O}(m)$.

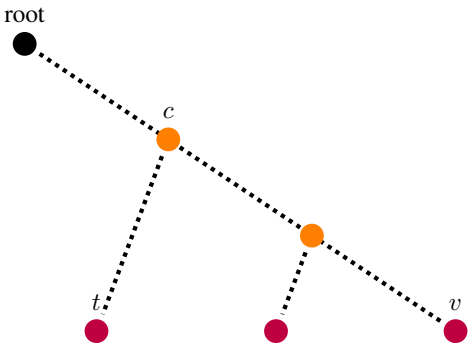

Fig. 2. For any ancestor c of v on the recursion tree $\mathcal{T}$, consider all nodes on the opposite branch of c where the SSSP instance involes vertex v. Then we can prove that all these nodes should form a tree path starting at c on the recursion tree $\mathcal{T}$. Ranging over different choices of c, all such nodes should form at most $O(\log n)$ tree paths, as drawn by the dotted lines, and so the total number of such nodes is bounded by $O(\log^2 n)$.

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*, pages 1681–1697. SIAM, 2014.

[2] Amir Abboud, Virginia Vassilevska Williams, and Joshua 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*, pages 377–391. SIAM, 2016.

[3] Boaz Ben-Moshe, Binay Bhattacharya, Qiaosheng Shi, and Arie Tamir. Efficient algorithms for center problems in cactus networks. *Theoretical Computer Science*, 378(3):237–252, 2007.

[4] Piotr Berman and Shiva Prasad Kasiviswanathan. Faster approximation of distances in graphs. In *Workshop on Algorithms and Data Structures*, pages 541–552. Springer, 2007.

[5] Shiri Chechik, Daniel H Larkin, Liam Roditty, Grant Schoenebeck, Robert E 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*, pages 1041–1052. SIAM, 2014.

[6] Victor Chepoi, Feodor Dragan, and Yann Vaxes. Center and diameter problems in plane triangulations and quadrangulations. In *SODA*, volume 2, pages 346–355. Citeseer, 2002.

[7] Fan RK Chung. Diameters of graphs: Old problems and new results. *Congressus Numerantium*, 60(2):295–317, 1987.

[8] Derek G Corneil, Feodor F Dragan, Michel Habib, and Christophe Paul. Diameter determination on restricted graph families. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 192–202. Springer, 1998.

[9] Lenore Cowen and Christopher G Wagner. Compact roundtrip routing for digraphs. In *Symposium on Discrete Algorithms: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, volume 17, pages 885–886. Citeseer, 1999.

[10] Mina Dalirrooyfard and Jenny Kaufmann. Approximation algorithms for min-distance problems in dags. *arXiv preprint arXiv:2106.02120*, 2021.

[11] Mina Dalirrooyfard, Virginia Vassilevska Williams, Nikhil Vyas, Nicole Wein, Yinzhan Xu, and Yuancheng Yu. Approximation algorithms for min-distance problems. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[12] 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.

[13] S Louis Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations research*, 12(3):450–459, 1964.

[14] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 515–524, 2013.

[15] Oren Weimann and Raphael Yuster. Approximating the diameter of planar graphs in near linear time. *ACM Transactions on Algorithms (TALG)*, 12(1):1–13, 2015.

[16] Christian Wulff-Nilsen. *Wiener index, diameter, and stretch factor of a weighted planar graph in subquadratic time*. Citeseer, 2008.

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