An Optimal-Time Construction of Sparse Euclidean Spanners with Tiny Diameter*

Shay Solomon†

Abstract

In STOC'95 [5] Arya et al. showed that for any set of n points in $\mathbb{R}^d$, a $(1+\epsilon)$-spanner with diameter at most 2 (respectively, 3) and $O(n \log n)$ edges (resp., $O(n \log\log n)$ edges) can be built in $O(n \log n)$ time. Moreover, Arya et al. [5] conjectured that one can build in $O(n \log n)$ time a $(1+\epsilon)$-spanner with diameter at most 4 and $O(n \log^* n)$ edges. Since then, this conjecture became a central open problem in this area. Nevertheless, very little progress on this problem was reported up to this date. In particular, the previous state-of-the-art subquadratic-time construction of $(1+\epsilon)$-spanners with $o(n \log\log n)$ edges due to Arya et al. [5] produces spanners with diameter 8.

In addition, general tradeoffs between the diameter and number of edges were established [5, 26]. Specifically, it was shown in [5, 26] that for any $k \geq 4$, one can build in $O(n(\log n)2^k \alpha_k(n))$ time a $(1+\epsilon)$-spanner with diameter at most $2k$ and $O(n2^k \alpha_k(n))$ edges. The function α_k is the inverse of a certain Ackermann-style function at the $\lfloor k/2 \rfloor$th level of the primitive recursive hierarchy, where $\alpha_0(n) = \lceil n/2 \rceil, \alpha_1(n) = \lceil \sqrt{n} \rceil, \alpha_2(n) = \lceil \log n \rceil, \alpha_3(n) = \lceil \log\log n \rceil, \alpha_4(n) = \log^* n, \alpha_5(n) = \lfloor \frac{1}{2} \log^* n \rfloor, \ldots$, etc. It is also known [26] that if one allows quadratic time then these bounds can be improved. Specifically, for any $k \geq 4$, a $(1+\epsilon)$-spanner with diameter at most k and $O(nk\alpha_k(n))$ edges can be constructed in $O(n^2)$ time [26]. A major open question in this area is whether one can construct within time $O(n \log n + nk\alpha_k(n))$ a $(1+\epsilon)$-spanner with diameter at most k and $O(nk\alpha_k(n))$ edges. This question in the particular case of $k = 4$ coincides with the aforementioned conjecture of Arya et al. [5].

In this paper we answer this long-standing question in the affirmative. Moreover, in fact, we provide a stronger result. Specifically, we show that for any $k \geq 4$, a $(1+\epsilon)$-spanner with diameter at most k and $O(n\alpha_k(n))$ edges can be built in optimal time $O(n \log n)$. In particular, our tradeoff for $k = 4$ provides an $O(n \log n)$-time construction of $(1+\epsilon)$-spanners with diameter at most 4 and $O(n \log^* n)$ edges, thus settling the conjecture of Arya et al. [5]. The tradeoff between the diameter and number of edges of our spanner construction is tight up to constant factors in the entire range of parameters, even if one allows the spanner to use (arbitrarily many) Steiner points.

1 Introduction

1.1 Euclidean Spanners. Consider a set S of n points in $\mathbb{R}^d$ and a number $t \geq 1$. A graph $G = (S, E)$ in which the weight $w(x, y)$ of each edge $e = (x, y) \in E$ is equal to the Euclidean distance $\|x - y\|$ between x and y is called a *geometric graph*. We say that the graph G is a *t-spanner* for S if for every pair $p, q \in S$ of distinct points, there exists a path in G between p and q whose weight (i.e., the sum of all edge weights in it) is at most t times the Euclidean distance $\|p - q\|$ between p and q. Such a path is called a *t-spanner path*. The problem of constructing Euclidean spanners has been studied intensively over the past two decades [14, 22, 4, 10, 15, 5, 16, 7, 28, 2, 11, 17, 31]. (See also the book by Narasimhan and Smid [26], and the references therein.) Also, Euclidean spanners find applications in geometric approximation algorithms, network topology design, geometric distance oracles, distributed systems, design of parallel machines, and other areas [15, 24, 28, 18, 20, 19, 21, 25].

Spanners are important geometric structures, since they enable approximation of the complete Euclidean graph in a much more economical form. First and foremost, a spanner should be *sparse*, meaning that it can have only a small (ideally, linear) number of edges. However, at the same time, the spanner is required to preserve some fundamental properties of the underlying complete graph. In particular, for some practical applications (e.g., in network routing protocols) it is desirable that the spanner achieves a small *diameter*, that is, for every pair $p, q \in S$ of distinct points there should be a t-spanner path that consists of a small number of edges [6, 1, 2, 11, 17].

In a seminal STOC'95 paper [5], Arya et al. showed

*Supported by the Clore Fellowship grant No. 81265410, by the BSF grant No. 2008430, and by the Lynn and William Frankel Center for Computer Science.

†Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel, shayso@cs.bgu.ac.il.

that for any set of n points in $\mathbb{R}^d$ one can build in $O(n \log n)$ time a $(1+\epsilon)$-spanner with diameter at most 2 and $O(n \log n)$ edges, and another one with diameter at most 3 and $O(n \log \log n)$ edges. Moreover, Arya et al. [5] conjectured[1] that one can build in $O(n \log n)$ time a $(1 + \epsilon)$-spanner with diameter at most 4 and $O(n \log^* n)$ edges. Since then, this conjecture became a central open problem in this area. Nevertheless, very little progress on this problem was reported up to this date. In particular, the previous state-of-the-art subquadratic-time construction of $(1 + \epsilon)$-spanners with $o(n \log \log n)$ edges due to Arya et al. [5] produces spanners with diameter 8.

In addition, general tradeoffs between the diameter and number of edges were established [5, 26]. Specifically, it was shown in [5, 26] that for any $k \geq 4$, one can build in $O(n(\log n)2^k \alpha_k(n))$ time a $(1+\epsilon)$-spanner with diameter at most $2k$ and $O(n2^k \alpha_k(n))$ edges. The function α_k is the inverse of a certain Ackermann-style function at the $\lfloor k/2 \rfloor$th level of the primitive recursive hierarchy, where $\alpha_0(n) = \lceil n/2 \rceil, \alpha_1(n) = \lceil \sqrt{n} \rceil, \alpha_2(n) = \lceil \log n \rceil, \alpha_3(n) = \lceil \log \log n \rceil, \alpha_4(n) = \log^* n, \alpha_5(n) = \lfloor \frac{1}{2} \log^* n \rfloor, \ldots$, etc. Roughly speaking, for $k \geq 2$ the function α_k is close to $\lfloor \frac{k-2}{2} \rfloor$-iterated log-star function, i.e., log with $\lfloor \frac{k-2}{2} \rfloor$ stars. (See Sect. 2 for the formal definition of this function.) Decreasing the diameter bound from $2k$ to k in the above tradeoff while maintaining the same running time appears to be difficult; indeed, no subquadratic-time construction of $(1 + \epsilon)$-spanners with diameter k and $O(n2^k \alpha_k(n))$ edges was reported since 1995. On the positive side, it is known [26] that if one allows quadratic time then these bounds can be improved. Specifically, for any $k \geq 4$, a $(1 + \epsilon)$-spanner with diameter at most k and $O(nk\alpha_k(n))$ edges can be constructed in $O(n^2)$ time [26].

A major open question in this area[2] is whether one can construct in time $O(n \log n + nk\alpha_k(n))$ a $(1 + \epsilon)$-spanner with diameter at most k and $O(nk\alpha_k(n))$ edges, for any $k \geq 4$. Notice that this question in the particular case of $k = 4$ coincides with the aforementioned conjecture of Arya et al. [5]. In this paper we answer this long-standing question in the affirmative. Moreover, in fact, we provide a stronger result. Specifically, we show that a $(1+\epsilon)$-spanner with diameter at most k and $O(n\alpha_k(n))$ edges can be built in optimal time $O(n \log n)$. In particular, our tradeoff for $k = 4$ provides an $O(n \log n)$-time construction of $(1+\epsilon)$-spanners with diameter at most 4 and $O(n \log^* n)$ edges, thus settling the conjecture of Arya et al. [5]. See Table 1 for a comparison of previous and new results for small values of diameter.

The complexity of this problem changes drastically if one settles for a relaxed bound of $\Lambda \cdot k$ rather than k on the diameter, for some large constant $\Lambda > 60$. Indeed, using classical constructions of 1-spanners for tree metrics [12, 34], building $(1 + \epsilon)$-spanners with diameter $\Lambda \cdot k$ and $O(n\alpha_k(n))$ edges in $O(n \log n)$ time is simple. (See Sect. 1.2 and Sect. 1.3 for further detail.) In fact, Chan and Gupta [11] devised such spanners in the more general setting of doubling metrics. Also, Solomon and Elkin [31] have recently devised a construction of sparse Euclidean spanners that achieves a tradeoff between the diameter, maximum degree, and weight. We remark that the construction of 1-spanners of [12] is embedded within both spanner constructions of [11] and [31]. A major drawback of the constructions of [11, 31] is that they cannot produce sparse spanners with diameter, say, at most 60.

Our tradeoff improves all previous results in a number of senses. In comparison to the construction of [26] that requires a quadratic running time, our construction is (1) drastically faster, and (2) produces a spanner that is sparser by a factor of k. In comparison to the result of [5, 26] that for any $k \geq 4$ produces in time $O(n(\log n)2^k \alpha_k(n))$, a $(1 + \epsilon)$-spanner with diameter at most $2k$ and $O(n2^k \alpha_k(n))$ edges, our construction has (1) a twice smaller diameter, (2) is faster by a factor of $2^k \alpha_k(n)$, and (3) produces a spanner that is sparser by a factor of 2^k. In comparison to the constructions of [11, 31], the diameter of our construction is smaller by a significant constant factor; in particular, our construction can produce spanners with diameter at most k, for any parameter $k \geq 4$, whereas the constructions of [11, 31] can only produce spanners whose diameter is no smaller than some large threshold value $\Lambda > 60$. See Table 2 for a comparison of previous and new tradeoffs.

In addition, substituting $k = 2\alpha(n) + 2$ in our tradeoff gives rise to a diameter of $2\alpha(n) + 2$ and $O(n \cdot \alpha_{2\alpha(n)+2}(n)) = O(n)$ edges, where $\alpha(\cdot)$ is the one-parameter inverse-Ackermann function. In all previous works of [5, 11, 26, 31], a construction of $(1 + \epsilon)$-spanners with diameter $O(\alpha(n))$ and $O(n)$ edges was also provided. However, the constants hidden within the O-notation of the diameter bound in the corresponding constructions of [5, 11, 26, 31] are no smaller than $\Lambda > 60$. Since $\alpha(n) \leq 4$ for all practical applications, this improvement on the diameter bound is significant.

Finally, Chan and Gupta [11] proved that there

[1] In fact, Arya et al. [5] explicitly claimed to have achieved an $O(n \log n)$-time construction of $(1 + \epsilon)$-spanners with diameter at most 4 and $O(n \log^* n)$ edges (see Theorem 8 in [5]). However, their proof turned out to be incorrect [29].

[2] It appears as open problem number 19 in the list of open problems in the treatise of Narasimhan and Smid [26] on Euclidean spanners; see p. 481. For a discussion on this problem we refer to p. 239 in [26].

	[5, 26]	[5, 26]	[11, 31]	[26]	[26]	**New**	**New**
Diameter	$4, 5, 6, 7$	8	$\Lambda > 60$	4	8	$\boldsymbol{4}$	$\boldsymbol{8}$
# edges	$O(n \log\log n)$	$O(n \log^* n)$	$O(n \log^* n)$	$O(n \log^* n)$	$O(n \log^{***} n)$	$\boldsymbol{O(n \log^* n)}$	$\boldsymbol{O(n \log^{***} n)}$
Time	$O(n \log n)$	$O(n(\log n) \log^* n)$	$O(n \log n)$	$O(n^2)$	$O(n^2)$	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(n \log n)}$

Table 1: A comparison of previous and new constructions of Euclidean $(1+\epsilon)$-spanners for small values of diameter. Our results are indicated by bold fonts. The letter Λ designates some large constant, $\Lambda > 60$; $\log^{***}$ denotes 3-iterated log-star function.

	[5, 26]	[11, 31]	[26]	**New**	[5, 11, 26, 31]	**New**
Diameter	$2k$	$\Lambda \cdot k$	k	$\boldsymbol{k}$	$\Lambda \cdot \alpha(n)$	$\boldsymbol{2\alpha(n) + 2}$
# edges	$O(n 2^k \alpha_k(n))$	$O(n \alpha_k(n))$	$O(nk\alpha_k(n))$	$\boldsymbol{O(n\alpha_k(n))}$	$\boldsymbol{O(n)}$	$\boldsymbol{O(n)}$
Time	$O(n(\log n) 2^k \alpha_k(n))$	$O(n \log n)$	$O(n^2)$	$\boldsymbol{O(n \log n)}$	$O(n \log n)$	$\boldsymbol{O(n \log n)}$

Table 2: A comparison of previous and new tradeoffs between diameter and number of edges of Euclidean $(1+\epsilon)$-spanners. Our tradeoff appears in the third column from the right, and is indicated by bold fonts. It is assumed that n and k are arbitrary parameters, with $k \geq 4$. The letter Λ designates some large constant, $\Lambda > 60$.
The two right-most columns compare previous and new constructions of $(1+\epsilon)$-spanners with $O(n)$ edges.

exists a set of n points on the x-axis for which any $(1+\epsilon)$-spanner with at most m edges must have a diameter at least $\alpha(m, n) - 4$, where $\alpha(\cdot,\cdot)$ is the two-parameter inverse-Ackermann function. This lower bound (cf. Corollary 4.1 therein [11]) implies that our tradeoff of k versus $O(n\alpha_k(n))$ between the diameter and number of edges cannot be improved[3] by more than constant factors even for 1-dimensional spaces. However, the lower bound of [11] does not preclude the existence of Euclidean Steiner spanners[4] with diameter $o(k)$ and $o(n\alpha_k(n))$ edges. In this paper we extend the lower bound of [11] to Euclidean Steiner spanners and show that as far as the diameter and number of edges are concerned, *Steiner points do not help.* Consequently, our construction of Euclidean spanners cannot be improved even if one allows the spanner to employ (arbitrarily many) Steiner points.

1.2 Spanners for Tree Metrics. The tree metric induced by an n-vertex (weighted) rooted tree (T, rt) is denoted by M_T. A spanning subgraph G of M_T is said to be a *1-spanner* for T, if for every pair of vertices, their distance in G is equal to their distance in T.

Alon and Schieber [3] and Bodlaender et al. [9] independently showed that for any n-point tree metric, a 1-spanner with diameter 2 (respectively, 3) and $O(n \log n)$ edges (resp., $O(n \log\log n)$ edges) can be built within time $O(n \log n)$ (resp., $O(n \log\log n)$). The constructions of [3] and [9] were also extended to higher *constant* values k of diameter, $k \geq 4$. Specifically, Alon and Schieber [3] showed that 1-spanners with diameter at most $2k$ (rather than k) and $O(n\alpha_k(n))$ edges can be built within $O(n\alpha_k(n))$ time. Bodlaender et al. [9] succeeded to construct 1-spanners with diameter at most k and $O(n\alpha_k(n))$ edges; however, the question of whether this construction of Bodlaender et al. can be implemented efficiently was left open in [9], and remained open prior to this work. Narasimhan and Smid [26] extended the constructions of [3] and [9] to super-constant values of k. Specifically, they showed that for any n-point tree metric and any $k \geq 4$, a 1-spanner with diameter at most $2k$ (respectively, k) and $O(n2^k\alpha_k(n))$ edges (resp., $O(nk\alpha_k(n))$ edges) can be built in time $O(n(\log n)2^k\alpha_k(n))$ (resp., $O(n^2)$).

There are also alternative constructions of 1-spanners for tree metrics [12, 34, 31]. However, these constructions produce spanners with diameter $\Lambda' \cdot k$ rather than $2k$ or k, for some large constant $\Lambda' > 30$. In particular, none of these constructions can produce a spanner whose diameter is, say, at most 30.

On the way to our results for Euclidean spanners, we have improved the constructions of [3, 9, 26] and devised an $O(n\alpha_k(n))$-time construction of 1-spanners for arbitrary n-point tree metrics with diameter at most k and $O(n\alpha_k(n))$ edges, for *any* $k \geq 4$. See Tables 3 and 4 for a comparison of previous and new results.

The running time of our construction is *linear* in the number of edges of the resulting spanners. Also, it was proved in [3] that any 1-spanner for the unweighted path graph P_n with diameter at most k must have at least $\Omega(n\alpha_k(n))$ edges. This lower bound implies that the tradeoff between the diameter and number of edges of our spanners is tight in the entire range of parameters.

We demonstrate that our construction of 1-spanners for tree metrics is useful for improving key results in

[3]See also open problem 20 on p. 481 in [26], and the corresponding solution [11, 30].

[4]A *Euclidean Steiner spanner* for a point set S is a spanner that may contain additional *Steiner points*, i.e., points that do not belong to the original point set S.

	[3]	[3]	[12, 34, 31]	[9, 26]	[9, 26]	**New**	**New**
Diameter	$4, 5, 6, 7$	8	$\Lambda' > 30$	4	8	**4**	**8**
# edges	$O(n \log \log n)$	$O(n \log^* n)$	$O(n \log^* n)$	$O(n \log^* n)$	$O(n \log^{***} n)$	$\boldsymbol{O(n \log^* n)}$	$\boldsymbol{O(n \log^{***} n)}$
Time	$O(n \log \log n)$	$O(n \log^* n)$	$O(n \log^* n)$	$O(n^2)$	$O(n^2)$	$\boldsymbol{O(n \log^* n)}$	$\boldsymbol{O(n \log^{***} n)}$

Table 3: A comparison of previous and new constructions of 1-spanners for arbitrary tree metrics for small values of diameter. Our results are indicated by bold fonts. Note that the constructions of [12, 34, 31] do not produce sparse spanners with diameter, say, at most 30. On the other hand, they can be employed to produce in time $O(n \log^* n)$, a 1-spanner with diameter at most Λ' and $O(n \log^* n)$ edges, where Λ' designates some large constant, $\Lambda' > 30$.

	[3, 26]	[12, 34, 31]	[9, 26]	**New**	[3, 12, 34, 26, 31]	**New**
Diameter	$2k$	$\Lambda' \cdot k$	k	$\boldsymbol{k}$	$\Lambda' \cdot \alpha(n)$	$\boldsymbol{2\alpha(n) + 2}$
# edges	$O(n 2^k \alpha_k(n))$	$O(n \alpha_k(n))$	$O(nk \alpha_k(n))$	$\boldsymbol{O(n \alpha_k(n))}$	$O(n)$	$\boldsymbol{O(n)}$
Time	$O(n (\log n) 2^k \alpha_k(n))$	$O(n \alpha_k(n))$	$O(n^2)$	$\boldsymbol{O(n \alpha_k(n))}$	$O(n)$	$\boldsymbol{O(n)}$

Table 4: A comparison of previous and new tradeoffs between diameter and number of edges of 1-spanners for arbitrary tree metrics. Our tradeoff appears in the third column from the right, and is indicated by bold fonts. It is assumed that n and k are arbitrary parameters, with $k \geq 4$. The letter Λ' designates some large constant, $\Lambda' > 30$.
The two right-most columns compare previous and new constructions of 1-spanners for tree metrics with $O(n)$ edges.

the context of Euclidean spanners. In addition, the problem of constructing 1-spanners for tree metrics is closely related to the well-studied partial-sums problem (see [33, 35, 13, 27], and the references therein). We anticipate that our construction of 1-spanners for tree metrics would be found useful in the context of the partial-sums problem, and for other applications such as those discussed in [9, 8].

1.3 Our and Previous Techniques. Our construction of sparse Euclidean spanners with bounded diameter employs a standard two-step scheme. First, build the *dumbbell trees* of Arya et al. [5]. Roughly speaking, the *Dumbell Theorem* of [5] states that for every set S of n points, one can efficiently construct a forest $\mathcal{F}$ of $O(1)$ rooted *dumbbell trees*, such that for any pair $p, q \in S$ of points, there is a dumbbell tree $T \in \mathcal{F}$, so that the path between p and q in T is a $(1 + \epsilon)$-spanner path. Second, build a "good" 1-spanner for each of the dumbbell trees of [5]. The union of all these 1-spanners yields our spanner construction.

This two-step scheme was used in all previous constructions of sparse spanners with bounded diameter for both Euclidean metrics [5, 26, 31] and doubling metrics [11].[5] Nevertheless, there is a fundamental difference between our and previous constructions. Specifically, to obtain "good" 1-spanners for the dumbbell trees of [5], all previous constructions of [5, 11, 26, 31] employed the classical constructions of 1-spanners for tree metrics of [12, 3, 9, 34] either directly or as a black box. In particular, Arya et al. [5], Chan and Gupta [11], and Narasimhan and Smid [26] employed directly the constructions of [3], [12], and [3, 9], respectively. Solomon and Elkin [31] employed the construction of [12] as a black box to obtain a new construction of sparse 1-spanners that achieves a tradeoff between the diameter, maximum degree, and weight; for unbounded values of maximum degree and weight, the construction of [31] reduces to that of [12]. On the other hand, in this paper we develop a novel construction of 1-spanners for tree metrics that improves upon the classical constructions of [3, 9]. By plugging our improved construction of 1-spanners on top of the dumbbell trees of [5], we obtain an improved construction of Euclidean spanners. Next, we discuss the technical challenges we faced on the way to achieving our optimal-time construction of 1-spanners for arbitrary n-point tree metrics with diameter at most k and $O(n \alpha_k(n))$ edges.

A central component in the constructions of 1-spanners for tree metrics of [12, 3, 9, 26, 31] is a tree decomposition procedure. Given an n-vertex rooted tree (T, rt) and a parameter ℓ, this procedure computes a set CV_ℓ of at most $O(n/\ell)$ cut vertices whose removal from the tree decomposes T into a collection of subtrees of size at most ℓ each. For our purposes, it is crucial that the running time of this procedure will be $O(n)$. Equally important, the size of the set CV_ℓ must not be greater (not even by a constant factor) than n/ℓ. None of the decomposition procedures of [12, 3, 9, 26, 31] satisfies these two requirements simultaneously. The decomposition procedure of [26], e.g., requires time $O(n \log n)$ rather than $O(n)$, and the size bound of CV_ℓ is $2(n/\ell)$ rather than n/ℓ; the slack of 2 on the size bound of CV_ℓ contributes a factor of 2^k to the

[5]Informally, instead of working with dumbbell trees, Chan and Gupta [11] used *net trees* that share similar properties.

bounds on the number of edges and the running time of the construction of [26]; also, the slack of $\log n$ on the running time of this procedure contributes an additional factor of $\log n$ to the bound on the running time of their construction; hence, the bounds on the number of edges and the running time of the construction of [26] are $O(n 2^k \alpha_k(n))$ and $O(n(\log n) 2^k \alpha_k(n))$, respectively. The decomposition procedure of [9] is the only one in which the size bound of CV_ℓ is at most n/ℓ, but it is unclear whether this procedure can be implemented efficiently. In this paper we devise a decomposition procedure that satisfies both these requirements. Our procedure is, in addition, surprisingly simple.

The main challenge that we faced was to achieve diameter k, rather than to settle for a diameter of at least $2k$ as is the case with all the previous subquadratic-time constructions of [12, 3, 34, 26, 31]. (Note that the aforementioned conjecture of Arya et al. [5] could not have been resolved with any bound larger than k at hand.) To understand where the difficulty lies, consider the set CV_ℓ of cut vertices. A key ingredient in the constructions of [12, 3, 9, 26] is the computation of an edge set E' that connects the cut vertices. In [3, 26], a natural yet inherently suboptimal approach was used, which leads to diameter $2k$ rather than k; first, compute a spanning tree T' over the vertex set $CV_\ell \cup \{rt\}$ that "inherits" the tree structure of the original rooted tree (T, rt) in the obvious way, and then recursively compute a 1-spanner for T'. Note that every 1-spanner path for T' between a pair of vertices, such that one is an ancestor of the other in T', is also a 1-spanner path for T. However, this property need not hold true for a general pair of vertices in T', since their *least common ancestor* in T might not be in T'. Consequently, we will get this way a 1-spanner for T with diameter $2k$ rather than k. (See Chapter 12 in [26] for further detail.) On the other hand, Chazelle [12] suggested connecting the vertices of CV_ℓ into a *Steiner tree* $\mathcal{T}^*$ using as many additional *Steiner vertices* as needed to guarantee that every 1-spanner path for $\mathcal{T}^*$ will also be a 1-spanner path for T. However, this approach is also doomed to failure. In particular, we mentioned above that it is critical to bound the number of cut vertices by n/ℓ; however, as it is impossible to distinguish between Steiner and non-Steiner vertices within the recursive call that computes a 1-spanner for T^*, the upper bound of n/ℓ should, in fact, apply to the total number of vertices in $\mathcal{T}^*$; alas, the number of vertices in T^* might generally grow far beyond n/ℓ. To overcome this obstacle, Bodlaender et al. [9] took the idea of [12] one step further and studied a generalized problem of constructing 1-spanners for *Steiner tree metrics*. Specifically, suppose that in T, a subset $R(T) \subseteq V(T)$ of the vertices are colored black, and the rest of the vertices in $S(T) = V(T) \setminus R(T)$ are colored white. The black (respectively, white) vertices are called the *required vertices* (resp., *Steiner vertices*) of T. We say that a 1-spanner H for T has diameter k if H contains a 1-spanner path for T that consists of at most k edges, for every pair of *required* vertices in T. This generalized setting seems to be fortuitous for our purposes; that is, now we "automatically" distinguish between Steiner and non-Steiner vertices within the recursive call that computes a 1-spanner for T^*, and so the upper bound of n/ℓ should no longer apply to the total number of vertices in T^*. However, it is much more difficult to provide a fast implementation when (possibly many) Steiner vertices join the game. In particular, Bodlaender et al. [9] provided a construction of 1-spanners for Steiner tree metrics with diameter at most k and $O(n\alpha_k(n))$ edges, for any *constant* $k \geq 4$; the previous state-of-the-art algorithm for implementing this construction requires quadratic time [26, 29]. To illustrate some of the difficulties that arise, consider two arbitrary Steiner trees T_1 and T_2 of the same required-size,[6] where T_1 has many more Steiner vertices than T_2. Clearly, it should take much more time to compute a 1-spanner for T_1 than for T_2. In other words, the running time should not only depend on the required-size of the tree, but also on its *Steiner-ratio*, defined as the ratio between the number of Steiner vertices and the number of required vertices in it. Suppose now that we are given a Steiner tree T of required-size n, and a set CV_ℓ of at most n/ℓ cut vertices that decomposes T into subtrees of required-size at most ℓ each. We would like to spend (roughly) the same amount of time within each recursive call of these subtrees; all these subtrees should thus have a similar Steiner-ratio, which should, in turn, be not (much) greater than the Steiner-ratio of the original tree T; alas, the number of Steiner vertices in a subtree might take any value from zero to the total number of Steiner vertices in T. The same problem occurs within the recursive call of the tree T^*, as its Steiner-ratio may be much greater than that of T. To tackle this problem we develop a linear time procedure for *pruning* the *redundant* vertices of a Steiner tree, while preserving its basic structure and intrinsic properties. We demonstrate that our tree pruning procedure guarantees that the Steiner-ratio in all the recursive calls will be smaller than 1. In addition, our pruning procedure provides an efficient method for computing the Steiner tree T^*. Ultimately, we produce an algorithm that implements the construction

[6]The *required-size* of a Steiner tree is defined as the number of required vertices in it.

of Bodlaender et al. [9] in *optimal time* $O(n\alpha_k(n))$, and, in addition, extends it to super-constant values of k.

1.4 Structure of the Paper. In Sect. 2 we present some inverse-Ackermann style functions that are used throughout the paper, and analyze their properties. Sect. 3 is devoted to our construction of 1-spanners for tree metrics. Therein we start (Sect. 3.1) with providing some relevant definitions. We proceed with presenting a tree pruning procedure (Sect. 3.2) and a tree decomposition procedure (Sect. 3.3). An optimal-time algorithm for computing 1-spanners for tree metrics is provided in Sect. 3.4. In Sect. 4 we derive our construction of Euclidean spanners. Finally, our lower bounds for Euclidean Steiner spanners are established in Sect. 5.

2 Some Inverse-Ackermann Style Functions

In this section we present some very slowly growing functions and analyze their properties.

Following [32, 3, 26], we define:

- $A_k(n) = A_{k-1}(A_k(n-1))$, for $k, n \geq 1$;
  $A_0(n) = 2n$, for $n \geq 0$; $A_k(0) = 1$, for $k \geq 1$.
- $B_k(n) = B_{k-1}(B_k(n-1))$, for $k, n \geq 1$;
  $B_0(n) = n^2$, for $n \geq 0$; $B_k(0) = 2$, for $k \geq 1$.

Also, we define the functional inverses of the functions A_k and B_k as follows:

- $\alpha_{2k}(n) = \min\{s \geq 0 : A_k(s) \geq n\}$, for $k, n \geq 0$.
- $\alpha_{2k+1}(n) = \min\{s \geq 0 : B_k(s) \geq n\}$, for $k, n \geq 0$.

We define $\log 0 = 0$. Note that for $n \geq 0$, $\alpha_0(n) = \lceil n/2 \rceil, \alpha_1(n) = \lceil \sqrt{n} \rceil, \alpha_2(n) = \lceil \log n \rceil, \alpha_3(n) = \lceil \log\log n \rceil, \alpha_4(n) = \log^* n, \alpha_5(n) = \lfloor \frac{1}{2} \log^* n \rfloor, \ldots$, etc. The following lemma can be easily verified.

Lemma 2.1. *(1) For $k \geq 0$, the function $\alpha_k = \alpha_k(n)$ is monotone non-decreasing with n. (2) For $k \geq 2$ and $n \geq 1$, $\alpha_k(n) < n$; for $n \geq 2$, $\alpha_0(n) < n$; for $n \geq 3$, $\alpha_1(n) < n$. (3) For $k, n \geq 0$, $\alpha_{k+2}(n) \leq \alpha_k(n)$.*

The following lemma from [26] provides a useful characterization of the function α_k.

Lemma 2.2. (Lemma 12.1.16 in [26], p. 230) *Let $k \geq 2$ be an arbitrary integer. Then $\alpha_k(n) = 1 + \alpha_k(\alpha_{k-2}(n))$, for $n \geq 3$, and $\alpha_k(0) = \alpha_k(1) = 0$. Also, $\alpha_k(2) = 0$ if k is odd, and $\alpha_k(2) = 1$ if k is even.*

Next, we define a variant α'_k of the function α_k.

- $\alpha'_0(n) = \alpha_0(n) = \lceil n/2 \rceil$, for $n \geq 0$;
  $\alpha'_1(n) = \alpha_1(n) = \lceil \sqrt{n} \rceil$, for $n \geq 0$.
- $\alpha'_k(n) = 2 + \alpha'_k(\alpha'_{k-2}(n))$, for $k \geq 2$ and $n \geq k+2$;
  $\alpha'_k(n) = \alpha_k(n)$, for $k \geq 2$ and $n \leq k+1$.

The next lemma establishes key properties of the function α'_k that will be used in the sequel.

Lemma 2.3. (Proof in Appendix A) *(1) For $k \geq 2$, the function $\alpha'_k = \alpha'_k(n)$ is monotone non-decreasing with n. (2) For $k \geq 2$ and $n \geq 1$, $\alpha'_k(n) < n$. (3) For $k \geq 2$ and $n \geq 0$, $\alpha'_{k+2}(n) \leq \alpha'_k(n)$.*

Note that for all $k, n \geq 0$, we have $\alpha'_k(n) \geq \alpha_k(n)$. Next, we argue that $\alpha'_k(n)$ is not much greater than $\alpha_k(n)$.

Lemma 2.4. (Proof in Appendix B) *For $k, n \geq 0$, $\alpha'_k(n) \leq 2\alpha_k(n) + 4$.*

The *Ackermann function* is defined by $A(n) = A_n(n)$, for $n \geq 0$, and the *one-parameter inverse-Ackermann function* is defined by $\alpha(n) = \min\{s \geq 0 : A(s) \geq n\}$, for $n \geq 0$. In [26] it is shown that $\alpha_{2\alpha(n)+2}(n) \leq 4$. (A similar bound was established in [23].) By Lemma 2.4, we get that $\alpha'_{2\alpha(n)+2}(n) \leq 12$. Finally, the *two-parameter inverse-Ackermann function* is defined by $\alpha(m, n) = \min\{s \geq 1 : A(s, 4\lceil m/n \rceil) \geq \log n\}$, for $m, n \geq 0$.

3 1-Spanners for (Steiner) Tree Metrics

3.1 Definitions and Notation. Let (T, rt) be an arbitrary n-vertex (weighted) rooted tree. Denote by $P_T(u, v)$ the unique path in T between a pair u, v of vertices in T. Let H be some *unweighted* spanning graph of $V(T)$. A path P in H between u and v is called T-monotone if it is a sub-path of $P_T(u, v)$, i.e., if $P_T(u, v) = (u = v_0, v_1, \ldots, v = v_t)$, then P can be written as $P = (u = v_{i_0}, v_{i_1}, \ldots, v_{i_q})$, where $0 = i_0 < i_1 < \ldots < i_q = t$. The *$T$-monotone distance* between a pair u, v of vertices in H is defined as the minimum number of edges in a T-monotone path in H between them. The *T-monotone diameter* of H, denoted $\Lambda_T(H) = \Lambda(H)$, is the maximum T-monotone distance between a pair of vertices in H. (If T is clear from the context, we may write diameter instead of T-monotone diameter.) By the triangle inequality, for any T-monotone path in H, the corresponding weighted path in the tree metric M_T induced by T is a 1-spanner path for T. Hence, H translates into a 1-spanner for T with diameter $\Lambda(H)$, and this holds true regardless of the actual weight function of T. We henceforth restrict attention to unweighted trees in the sequel.

Following [9], we study a generalization of the problem for *Steiner trees*, where there is a designated subset $R(T) \subseteq V(T)$ of *required vertices*, and the *diameter* of a 1-spanner for T is defined as the maximum T-monotone distance between a pair of *required* vertices.

The remaining vertices in $S(T) = V(T) \setminus R(T)$ are called the *Steiner vertices* of T. We define the *required-size* (respectively, *Steiner-size*) of T as the number of required (resp., Steiner) vertices in it. Also, the *Steiner-ratio* of T is defined as the ratio between the Steiner-size and the required-size of T. If the number of Steiner vertices in T is (much) larger than the number of required vertices, or equivalently, if the Steiner-ratio of T is (much) larger than 1, it might be possible to *prune* some *redundant* Steiner vertices from T while preserving its basic structure. A Steiner tree T' is called *T-monotone preserving*, if (1) $R(T') = R(T)$, and (2) for every pair u, v of required vertices, the unique path $P_{T'}(u, v)$ between u and v in T' is T-monotone. Consider a T-monotone preserving tree T', and let u, v be an arbitrary pair of required vertices. Note that any T'-monotone path between u and v is also T-monotone. Thus any 1-spanner H' for T' with T'-monotone diameter k is also a 1-spanner for T with T-monotone diameter k.

Finally, for a positive integer z, we denote the set $\{1, 2, \ldots, z\}$ by $[z]$.

3.2 Tree Pruning Procedure. In this section we devise a linear time procedure for pruning the redundant vertices of a Steiner tree while preserving its basic structure and intrinsic properties.

For a Steiner rooted tree (T, rt) and a pair u, v of vertices in T, let $LCA_T(u, v)$ denote the *least common ancestor* (henceforth, LCA) of u and v in T. A Steiner vertex $x \in S = S(T)$ in T is called *useful* if it is the LCA of some pair of required vertices $u, v \in R = R(T)$. Otherwise it is called *redundant.* Denote by $LCA(T)$ the set of all useful vertices in T, i.e., $LCA(T) = \{x \in S \mid \exists u, v \in R : x = LCA_T(u, v)\}$. A Steiner rooted tree with no redundant vertices is called *pruned.* We denote the children of the root vertex rt in a Steiner rooted tree (T, rt) by $c_1, \ldots, c_{ch(rt)}$, where $ch(rt)$ designates the number of children of rt in T. For each index $i \in [ch(rt)]$, let $T_{(i)}$ be the subtree of T rooted at c_i. We say that the subtree $T_{(i)}$ is *required* if it contains at least one required vertex, i.e., if $R_{(i)} = R \cap V(T_{(i)})$ is non-empty. Otherwise we say that it is *redundant.* Note that all vertices in a redundant subtree are redundant. Denote by $I = I(T)$ the set of all indices i, such that $i \in [ch(rt)]$ and $T_{(i)}$ is a required subtree.

Next, we present a procedure $Prune = Prune((T, rt))$ that accepts as input a Steiner rooted tree (T, rt), and transforms it into a pruned T-monotone preserving tree (T', rt').

If T consists of just the single vertex rt, then the procedure either leaves T intact if $rt \in R$, or it transforms T into an empty tree if $rt \notin R$. Otherwise,

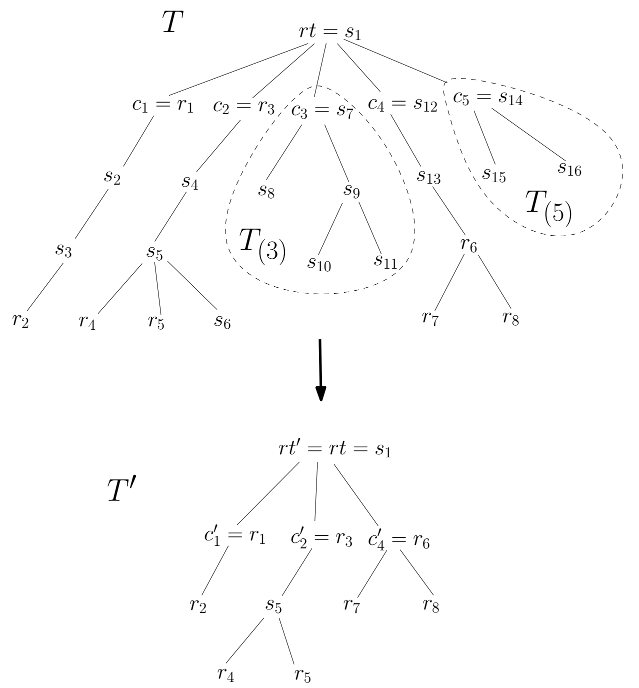

Figure 1: A rooted Steiner tree (T, rt) is depicted at the top of the figure, having 8 required vertices $r_1, \ldots, r_8$ and 16 Steiner vertices $rt = s_1, \ldots, s_{16}$. The two subtrees $T_{(3)}$ and $T_{(5)}$ of T that are depicted within dashed lines are redundant, whereas the other three subtrees $T_{(1)}, T_{(2)}$, and $T_{(4)}$ of T are required. The pruned T-monotone preserving tree (T', rt') depicted at the bottom is obtained as a result of the invocation of the procedure $Prune$ on T.

$|T| \geq 2$. For each $i \in [ch(rt)]$, the tree $(T_{(i)}, c_i)$ is recursively transformed into a pruned $T_{(i)}$-monotone preserving tree $(T'_{(i)}, c'_i)$. Observe that for each $i \in [ch(rt)] \setminus I$, $R_{(i)} = \emptyset$ and $T_{(i)}$ is a redundant subtree, and so $T'_{(i)}$ is empty. Also, for each $i \in I$, the subtree $T'_{(i)}$ is non-empty. The procedure removes all $[ch(rt)]$ edges that connect rt with its children in T. The execution of the procedure then splits into four cases.

Case 1: $rt \in R$. The root vertex rt of T remains the root vertex of T', and for each index $i \in [I]$, an edge connecting rt with the root c'_i of $T'_{(i)}$ is added.

Case 2: $rt \notin R$ and $I = \emptyset$. Hence, $R = \emptyset$, and the procedure transforms T into an empty tree.

Case 3: $rt \notin R$ and $|I| = 1$. In this case rt is redundant, and there is a single non-empty subtree $T'_{(p)}$, i.e., $I = \{p\}$, for some index $p \in [ch(rt)]$. Hence, the procedure removes rt and sets $T' = T'_{(p)}$.

Case 4: $rt \notin R$ and $|I| \geq 2$. In this case rt is useful. As in case 1, the root vertex rt of T remains the root vertex of T', and for each index $i \in [I]$, an edge connecting rt with the root c'_i of $T'_{(i)}$ is added.

(See Fig. 1 for an illustration.)

It is easy to verify that the procedure $Prune$ can be implemented in linear time. Next, we analyze the properties of the resulting tree T'.

The next lemma follows easily from the description of the procedure.

LEMMA 3.1. *(T', rt') is a Steiner rooted tree over $V(T') = R(T) \cup LCA(T)$, and $R(T') = R(T)$. Also, for each $i \in I$, $(T'_{(i)}, c'_i)$ is a Steiner rooted tree over $V(T'_{(i)}) = R(T_{(i)}) \cup LCA(T_{(i)})$, and $R(T'_{(i)}) = R(T_{(i)})$.*

LEMMA 3.2. *For any pair u, v of vertices in T', u is an ancestor of v in T' iff it is its ancestor in T.*

Proof. The proof is by induction on $n' = |T'|$. The basis $n' \leq 1$ holds vacuously.
Induction Step: We assume the correctness of the statement for all smaller values of n', $n' \geq 2$, and prove it for n'. Since $n' \geq 2$, it must hold that $|I| \geq 1$. Next, we prove the "only if" part. The argument for the "if" part is similar. Consider an arbitrary pair u, v of vertices in T', such that u is an ancestor of v in T'. Next, we show that u is also an ancestor of v in T. By Lemma 3.1, for each index $i \in I$, $V(T'_{(i)}) \subseteq V(T_{(i)})$. The analysis splits into two cases.
Case 1: $|I| = 1$ and $rt \notin R$. In this case $T' = T'_{(p)}$, with $I = \{p\}$. Notice that u and v belong to $T_{(p)}$. By the induction hypothesis for $T'_{(p)}$, u is an ancestor of v in $T_{(p)}$, and thus also in T.
Case 2: Either $rt \in R$ or $|I| \geq 2$. In both cases $rt(T') = rt$, and for each index $i \in I$, the root c'_i of the subtree $T'_{(i)}$ is a child of rt in T'.
If both u and v belong to the same subtree $T'_{(i)}$, for some index $i \in I$, then they both belong to $T_{(i)}$. Hence, by the induction hypothesis for $T'_{(i)}$, u is an ancestor of v in $T_{(i)}$, and thus also in T.
Since u is an ancestor of v in T', u and v cannot belong to different subtrees $T'_{(j)}$ and $T'_{(k)}$ of T', $j, k \in I$. Hence, the remaining case is that $u = rt(T') = rt$. Clearly, rt is an ancestor of v in T, and we are done. ∎

LEMMA 3.3. *For any pair u, v of required vertices, $LCA_{T'}(u, v) = LCA_T(u, v)$.*

Proof. Write $l' = LCA_{T'}(u, v)$ and $l = LCA_T(u, v)$. Note that l is either a required vertex or a useful vertex. By Lemma 3.1, l belongs to T'. By definition, l' is the LCA of u and v in T'. By Lemma 3.2, we get that l' is a common ancestor of u and v in T, and so it must be an ancestor of their LCA l in T. Lemma 3.2 implies that l' is an ancestor of l also in T'. However, by applying Lemma 3.2 again, we conclude that l is a common ancestor of u and v in T', and so it must be an ancestor of their LCA l' in T'. Hence, $l' = l$. ∎

Lemmas 3.1 and 3.3 yield the following corollary.

COROLLARY 3.1. *(T', rt') is pruned.*

Proof. We argue that $LCA(T') = LCA(T)$. Indeed, by Lemma 3.1, $V(T') = R(T) \cup LCA(T)$ and $R(T') = R(T)$. Hence, $S(T') = LCA(T)$, and so $LCA(T') \subseteq S(T') = LCA(T)$. To see why $LCA(T) \subseteq LCA(T')$ holds true as well, consider a vertex $l \in LCA(T)$. By definition, $l \notin R(T)$, and there exists a pair of required vertices $u, v \in R(T)$, such that $l = LCA_T(u, v)$. Hence, $l \notin R(T')$, and by Lemma 3.3, $l = LCA_{T'}(u, v)$. It follows that $l \in LCA(T')$.

Consequently, $V(T') = R(T') \cup LCA(T')$, and so there are no redundant vertices in T'. ∎

LEMMA 3.4. *For any pair u, v of vertices, such that u is an ancestor of v in T', $P_{T'}(u, v)$ is T-monotone.*

Proof. Write $P_{T'}(u, v) = (u = v_0, v_1, \ldots, v = v_q)$. By Lemma 3.2, for each index $i \in [q]$, v_{i-1} is an ancestor of v_i in T. Hence, $P_{T'}(u, v)$ is a sub-path of $P_T(u, v)$, or equivalently, it is T-monotone. ∎

We conclude that T' is T-monotone preserving.

COROLLARY 3.2. *For any pair u, v of required vertices, $P_{T'}(u, v)$ is T-monotone.*

Proof. If u is either an ancestor or a descendant of v in T', then the statement follows from Lemma 3.4.

We may henceforth assume that $LCA_{T'}(u, v) \neq u, v$. Write $l' = LCA_{T'}(u, v)$ and $l = LCA_T(u, v)$. By Lemma 3.3, $l' = l$. Observe that $P_{T'}(u, v)$ is a concatenation of the two paths $P_{T'}(u, l)$ and $P_{T'}(l, v)$, i.e., $P_{T'}(u, v) = P_{T'}(u, l) \circ P_{T'}(l, v)$. Similarly, we have $P_T(u, v) = P_T(u, l) \circ P_T(l, v)$. Lemma 3.4 implies that both $P_{T'}(u, l)$ and $P_{T'}(l, v)$ are T-monotone, or equivalently, $P_{T'}(u, l)$ is a sub-path of $P_T(u, l)$ and $P_{T'}(l, v)$ is a sub-path of $P_T(l, v)$. It follows that $P_{T'}(u, v) = P_{T'}(u, l) \circ P_{T'}(l, v)$ is a sub-path of $P_T(u, v) = P_T(u, l) \circ P_T(l, v)$, i.e., $P_{T'}(u, v)$ is T-monotone. ∎

Having proved that T' is a pruned T-monotone preserving tree, we turn to establish a number of basic properties of pruned trees that will be used in the sequel.

A Steiner tree in which the number of Steiner vertices is smaller than the number of required vertices is called *compact*. (By definition, the Steiner-ratio of a compact tree is smaller than 1.)

Note that in any non-empty pruned tree T, $R \neq \emptyset$ and $S = LCA(T)$. The next lemma implies that any non-empty pruned tree is compact.

LEMMA 3.5. *For any Steiner rooted tree (T, rt) (not necessarily pruned), $|LCA(T)| \leq \max\{0, |R| - 1\}$.*

Proof. The proof is by induction on $n = |T|$. The basis $n = 0$ is trivial.

Induction Step: We assume the correctness of the statement for all smaller values of n, $n \geq 1$, and prove it for n. If $R = \emptyset$, then by definition $LCA(T) = \emptyset$ as well, and we are done.

We henceforth assume that R is non-empty, and so $\max\{0, |R| - 1\} = |R| - 1$. By definition, for each $i \in I$, $R_{(i)} \neq \emptyset$, and for each $i \in [ch(rt)] \setminus I$, $R_{(i)} = \emptyset$. Hence, by the induction hypothesis, for each $i \in I$, $|LCA(T_{(i)})| \leq \max\{0, |R_{(i)}| - 1\} = |R_{(i)}| - 1$, and for each $i \in [ch(rt)] \setminus I$, $LCA(T_{(i)}) = \emptyset$. Clearly, the sets $\{R_{(i)}\}_{i \in I}$ and $\{LCA(T_{(i)})\}_{i \in I}$ are pairwise disjoint.

The analysis splits into three cases.

Case 1: $rt \in R$. In this case $R = \bigcup_{i \in I} R_{(i)} \cup \{rt\}$ and $LCA(T) = \bigcup_{i \in I} LCA(T_{(i)})$, implying that $|R| = \sum_{i \in I} |R_{(i)}| + 1$ and $|LCA(T)| = \sum_{i \in I} |LCA(T_{(i)})|$. Altogether,

$$\begin{aligned} |LCA(T)| &= \sum_{i \in I} |LCA(T_{(i)})| \leq \sum_{i \in I} \left(|R_{(i)}| - 1\right) \\ &= \sum_{i \in I} |R_{(i)}| - |I| \leq |R| - 1. \end{aligned}$$

Case 2: rt is redundant, i.e., $rt \in S \setminus LCA(T)$. Since $R \neq \emptyset$ and rt is redundant, it must hold that $|I| = 1$, i.e., $I = \{p\}$, for some index $p \in [ch(rt)]$. Hence, $R = R_{(p)}$ and $LCA(T) = LCA(T_{(p)})$, implying that $|LCA(T)| = |LCA(T_{(p)})| \leq |R_{(p)}| - 1 = |R| - 1$.

Case 3: rt is useful, i.e., $rt \in LCA(T)$. In this case there must be at least two different required subtrees $T_{(j)}$ and $T_{(k)}$, $j, k \in I$, and so $|I| \geq 2$. Observe that $R = \bigcup_{i \in I} R_{(i)}$ and $LCA(T) = \bigcup_{i \in I} LCA(T_{(i)}) \cup \{rt\}$, implying that $|R| = \sum_{i \in I} |R_{(i)}|$ and $|LCA(T)| = \sum_{i \in I} |LCA(T_{(i)})| + 1$. It follows that

$$\begin{aligned} |LCA(T)| &= \sum_{i \in I} |LCA(T_{(i)})| + 1 \\ &\leq \sum_{i \in I} \left(|R_{(i)}| - 1\right) + 1 \leq |R| - 1. \end{aligned}$$ ∎

In what follows we denote the number of edges in a path P by $|P|$.

Lemma 3.6. *For a non-empty pruned tree (T, rt), its depth $h(T)$ is at most $|R| - 1$ and its T-monotone diameter $\Lambda(T)$ is at most $|R|$. Moreover, $\Lambda(T)$ is equal to $|R|$ only if the following conditions hold:*
(1) $rt \notin R$, (2) rt has exactly two children, and (3) For any pair u, v of vertices in T for which $|P_T(u, v)| = |R|$, it holds that $u, v \neq rt = LCA_T(u, v)$.

Proof. The proof is by induction on $n = |T|$. The basis $n = 1$ is trivial.

Induction Step: We assume the correctness of the statement for all smaller values of n, $n \geq 2$, and prove it for n. Since T is pruned, all the subtrees $T_{(1)}, \ldots, T_{(ch(rt))}$ of T are pruned as well, and so the induction hypothesis applies to every one of them. Fix an arbitrary index $i \in [ch(rt)]$. Since $T_{(i)}$ is pruned, we have $|R_{(i)}| \geq 1$. We argue that $|R_{(i)}| \leq |R| - 1$. This is clearly the case if $rt \in R$. Otherwise, rt must be useful, and so it must have at least two children in T. Hence, there is another index $j \in [ch(rt)]$, such that $|R_{(j)}| \geq 1$. Since $R_{(i)} \cup R_{(j)} \subseteq R$, we get that $|R_{(i)}| \leq |R| - |R_{(j)}| \leq |R| - 1$.

By the induction hypothesis, for each $i \in [ch(rt)]$, $h(T_{(i)}) \leq |R_{(i)}| - 1 \leq |R| - 2$. Hence,

$$h(T) = \max\{h(T_{(i)}) \mid i \in [ch(rt)]\} + 1 \leq |R| - 1.$$

To bound the diameter $\Lambda(T)$ of T, consider a pair u, v of vertices in T for which $|P_T(u, v)| = \Lambda(T)$. If u and v belong to the same subtree $T_{(i)}$ of T, for some $i \in [ch(rt)]$, then $\Lambda(T) = \Lambda(T_{(i)})$, and by the induction hypothesis for $T_{(i)}$, $\Lambda(T) = \Lambda(T_{(i)}) \leq |R_{(i)}| \leq |R| - 1$. Otherwise, $rt = LCA_T(u, v)$. If either u or v is the root vertex rt, then $\Lambda(T) \leq h(T) \leq |R| - 1$.

So far we have proved that in order to obtain $\Lambda(T) \geq |R|$, it must hold that $u, v \neq rt = LCA_T(u, v)$. We henceforth assume that $u, v \neq rt = LCA_T(u, v)$. In other words, u and v belong to different subtrees $T_{(i)}$ and $T_{(j)}$ of T, respectively, for some indices $i, j \in [ch(rt)]$. Observe that $|P_T(u, v)| \leq h(T_{(i)}) + h(T_{(j)}) + 2$. By the induction hypothesis for $T_{(i)}$ and $T_{(j)}$, $h(T_{(i)}) \leq |R_{(i)}| - 1$ and $h(T_{(j)}) \leq |R_{(j)}| - 1$, and so $|P_T(u, v)| \leq |R_{(i)}| + |R_{(j)}|$. It follows that $\Lambda(T) = |P_T(u, v)| \leq |R_{(i)}| + |R_{(j)}| \leq |R|$. Moreover, one can have $\Lambda(T) = |R|$ only if $|R_{(i)}| + |R_{(j)}| = |R|$, in which case both $rt \notin R$ and $ch(rt) = 2$ must hold. ∎

Corollary 3.3. *Let (T, rt) be a pruned tree, such that rt has exactly two children c_1 and c_2, and let $\tilde{T}$ be the graph obtained from T by adding to it the edge (c_1, c_2). Then the T-monotone diameter $\Lambda(\tilde{T})$ of $\tilde{T}$ is at most $|R| - 1$.*

Proof. Consider a pair u, v of vertices in $\tilde{T}$ for which their T-monotone distance δ satisfies $\Lambda(\tilde{T}) = \delta$. Since $\tilde{T}$ contains all edges of T, we have $\delta \leq |P_T(u, v)|$. If $|P_T(u, v)| \leq |R| - 1$, then we are done. Otherwise, by Lemma 3.6, $|P_T(u, v)| = |R|$ and $u, v \neq rt = LCA_T(u, v)$. Hence, either u belongs to $T_{(1)}$ and v belongs to $T_{(2)}$, or vice versa. Suppose without loss of generality that u belongs to $T_{(1)}$ and v belongs to $T_{(2)}$, and write $P_T(u, v) = (u = v_0, v_1, \ldots, c_1 = v_{j-1}, rt = v_j, c_2 = v_{j+1}, v_{j+2}, \ldots, v = v_{|R|})$. Notice that $\tilde{T}$ contains all edges of $P_T(u, v)$, and, in addition, the edge (c_1, c_2), which can be used as a shortcut to avoid the detour (c_1, rt, c_2) around rt. It follows that $\tilde{T}$ contains the T-monotone path $\tilde{P} = (u = v_0, v_1, \ldots, c_1 = v_{j-1}, c_2 = v_{j+1}, v_{j+2}, \ldots, v = v_{|R|})$ that consists of $|R| - 1$ edges, and so $\Lambda(\tilde{T}) = \delta \leq |\tilde{P}| = |R| - 1$. ∎

3.3 Tree Decomposition Procedure. In this section we devise a linear time procedure for decomposing a Steiner tree into subtrees in an optimal way.

For a Steiner rooted tree (T, rt) and a subset C of $V(T)$, we denote by $T \setminus C$ the forest obtained from T by removing from it all the vertices in C along with the edges that are incident to them.

Next, we present a procedure $\mathit{Decomp} = \mathit{Decomp}((T, rt), \ell)$ that accepts as input a Steiner rooted tree (T, rt) with required-size n and a positive integer ℓ, and returns as output a subset $CV_\ell \subseteq V(T)$ of *cut vertices*. We do not require that a cut vertex would belong to $R = R(T)$.

For each vertex v in T we hold a variable $\mathit{size}(v)$. Also, we initialize the set CV_ℓ to $\emptyset$. The procedure visits the vertices of T in a post-order manner, so that a vertex v is visited only after all its children have been visited. For each visited vertex v, the procedure assigns $\mathit{size}(v) = 1 + \sum_{u \in N(v)} \mathit{size}(u)$ if $v \in R$, and $\mathit{size}(v) = \sum_{u \in N(v)} \mathit{size}(u)$ otherwise, where $N(v)$ denotes the set of children of v in the (current) tree T. (If v is a leaf, then $N(v) = NULL$, and so $\mathit{size}(v) = 1$ if $v \in R$, and $\mathit{size}(v) = 0$ otherwise.) Also, if $\mathit{size}(v) > \ell$, the procedure designates v as a cut vertex by inserting it to CV_ℓ, and then removes the subtree T_v of T rooted at v from T. (See Fig. 2 for an illustration.)

Note that the running time of the procedure Decomp is linear in the number of vertices in T. In particular, if T is pruned, we get a running time of $O(n)$.

Next, observe that at the end of the execution of the procedure Decomp, for any subtree $\tau \in T \setminus CV_\ell$ and any vertex $w \in \tau$, $\mathit{size}(w)$ holds the required-size of the subtree τ_w of τ rooted at w, i.e., $\mathit{size}(w) = |R(\tau_w)|$.

The following lemma establishes upper bounds on the maximum required-size of a subtree in $T \setminus CV_\ell$ and the size of the set CV_ℓ of cut vertices that is returned by the procedure Decomp.

LEMMA 3.7. *(1) The required-size $|R(\tau)|$ of any subtree $\tau \in T \setminus CV_\ell$ is at most ℓ. (2) $|CV_\ell| \leq \lfloor \frac{n}{\ell+1} \rfloor$.*

Proof. (1) Consider an arbitrary subtree τ in $T \setminus CV_\ell$, and let x be the root vertex of τ. By the description of the procedure and the above observation, we have $|R(\tau)| = |R(\tau_x)| = \mathit{size}(x) \leq \ell$, as otherwise x would have been designated as a cut vertex. (2) Immediately after a cut vertex v is inserted into CV_ℓ, the procedure removes the subtree T_v of T rooted at v from T, and so the required-size of the tree T is being decreased by $|R(T_v)|$ units. Define $\chi(v) = 1$ if $v \in R$, and $\chi(v) = 0$ otherwise. By the description of the procedure and the above observation, just before the removal of T_v from T we have

$$\begin{aligned} |R(T_v)| &= \chi(v) + \sum_{u \in N(v)} |R(T_u)| \\ &= \chi(v) + \sum_{u \in N(v)} \mathit{size}(u) \;=\; \mathit{size}(v) \;>\; \ell, \end{aligned}$$

implying that the required-size of T is being decreased by at least $\ell + 1$ units. Hence, after i cut vertices have been inserted into CV_ℓ, the required-size of T is at most $n - i(\ell + 1)$. Also, from the moment the required-size of T becomes at most ℓ, the set CV_ℓ remains intact. ∎

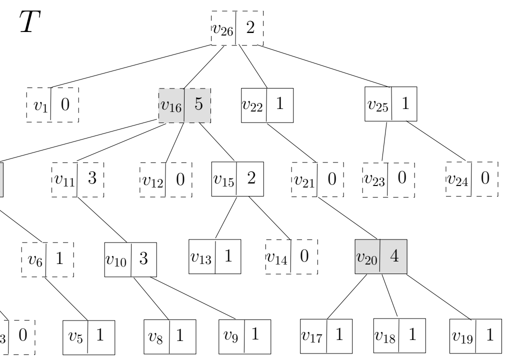

Figure 2: An illustration of a rooted Steiner tree (T, rt) over the vertices $v_1, v_2, \ldots, v_{26}$, where $rt = v_{26}$, and for each $i \in [26]$, v_i is the ith visited vertex in a (left-to-right) post-order traversal. Each vertex v_i in the tree is represented as a two-cell rectangle, with the left cell holding its name v_i, and the right one holding the value of $\mathit{size}(v_i)$. The 15 required vertices of the tree are depicted with solid lines, whereas the 11 Steiner vertices are depicted with dashed lines. The three vertices whose bounding rectangles are filled, i.e., v_7, v_{16} and v_{20}, comprise the set CV_3 of cut vertices that is returned as output of the call $\mathit{Decomp}((T, rt), \ell = 3)$.

Remark: The tradeoff ℓ versus $\lfloor \frac{n}{\ell+1} \rfloor$ between the maximum required-size of a subtree in $T \setminus CV_\ell$ and the size of the set CV_ℓ of cut vertices is tight, and is realized when T is the unweighted path graph P_n.

3.4 Sparse 1-Spanners for (Steiner) Tree Metrics with Bounded Diameter. In this section we present an optimal-time algorithm for computing sparse 1-spanners for Steiner tree metrics with bounded diameter. The tradeoff between the diameter and number of edges of our spanners is tight up to constant factors.

Let (T, rt) be a Steiner rooted tree. Notice that T can be transformed in linear time into a pruned T-monotone preserving tree (T', rt') by invoking the procedure *Prune* described in Sect. 3.2 on T. Also, any 1-spanner for T' provides a 1-spanner for the original tree T with the same diameter. We henceforth assume that the original tree T is pruned, i.e., $T = T'$. We also assume that for each vertex v in T, it can be decided in constant time whether it is a required or a Steiner vertex, i.e., whether $v \in R(T)$ or $v \in S(T)$.

Next, we describe an algorithm $Tree1Spanner = Tree1Spanner((T, rt), n, k)$ that accepts as input a pruned tree (T, rt), an integer $n \geq 0$ that designates the required-size of T, and an integer $k \geq 2$, and returns as output a 1-spanner for T.

If $0 \leq n \leq k$, return the edge set $E(T)$ of T.

If $n = k + 1$, check whether rt has exactly two children. If so, return $E(T) \cup \{(c_1, c_2)\}$, where c_1 and c_2 designate these two children. Otherwise, return $E(T)$.

We henceforth assume that $n \geq k + 2$.

The execution of the algorithm splits into six steps.

- At the first step, set $\ell = \alpha'_{k-2}(n)$, and compute the set CV_ℓ of cut vertices of T by calling $Decomp((T, rt), \ell)$.
- At the second step, compute the edge set E' that connects the cut vertices. If $k = 2$, set $E' = \emptyset$. If $k = 3$, set E' as the edge set of the complete graph over CV_ℓ. For $k \geq 4$, proceed as follows. (1) Compute a copy τ of T. (2) Go over all the vertices of τ and designate the vertices of CV_ℓ as the required vertices of τ. (Thus $R(\tau) = CV_\ell$, and $S(\tau) = V(T) \setminus CV_\ell$.) (3) Compute the pruning τ' of τ by making the call $Prune((\tau, rt))$. (4) Set E' as the edge set returned by the recursive call $Tree1Spanner((\tau', rt(\tau')), |CV_\ell|, k - 2)$.
- At the third step, compute the subtrees $T_1, \ldots, T_g$ in $T \setminus CV_\ell$.
- At the fourth step, compute the edge set E'' that connects the cut vertices with the corresponding subtrees. Specifically, the set of all cut vertices $u \in CV_\ell$ that are connected by an edge of T to some vertex of T_i is called the *border* of T_i, for each $i \in [g]$. The vertex u is called a *border vertex* of T_i. Compute the edge set $E'' = \{(u, v) \mid v \in R(T) \setminus CV_\ell, u \in CV_\ell, u \text{ is a border vertex of the subtree to which } v \text{ belongs}\}$.

(See Fig. 3 for an illustration.)

- At the fifth step we would like to proceed recursively for each of the subtrees $T_1, \ldots, T_g$. To this end, first compute the pruning T'_i of T_i by making the call $Prune((T_i, rt(T_i)))$, for each $i \in [g]$, and then set E_i to be the edge set returned by the recursive call $Tree1Spanner((T'_i, rt(T'_i)), |R_i|, k)$, where $R_i = R(T_i)$.
- Finally, at the sixth step, return the edge set $E = E' \cup E'' \cup \bigcup_{i=1}^{g} E_i$.

The following theorem summarizes the properties of Algorithm $Tree1Spanner((T, rt), n, k)$.

THEOREM 3.1. *Let $k \geq 2$ and $n \geq 0$ be two arbitrary integers, and let (T, rt) be an arbitrary pruned tree with required-size n. Algorithm $Tree1Spanner((T, rt), n, k)$ computes in time $O(n\alpha_k(n))$ a 1-spanner $G_T = (V(T), E)$ for T, having diameter at most k and $O(n\alpha_k(n))$ edges.*

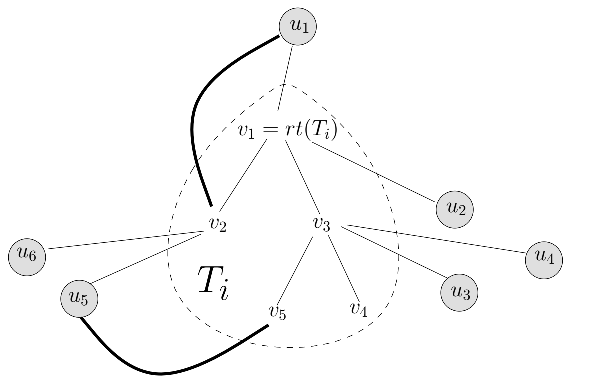

Figure 3: An illustration of a subtree $T_i \in T \setminus CV_\ell$ that contains the five vertices $rt(T_i) = v_1, v_2, \ldots, v_5$, with v_3 being the only Steiner vertex in T_i. The border of T_i is comprised of the six vertices $u_1, u_2, \ldots, u_6$ that are depicted within filled circles. Each required vertex v_j in T_i, $j \in [5], j \neq 3$, is incident on the single upstream edge (u_1, v_j), and on the five downstream edges $(u_2, v_j), (u_3, v_j), \ldots, (u_6, v_j)$. The upstream edge (u_1, v_2) and the downstream edge (u_5, v_5) are depicted by bold lines. (See the proof of Lemma 3.8 for the definitions of *upstream* and *downstream* edges.)

Remarks: (1) If we set $\ell = \alpha_{k-2}(n)$ instead of $\ell = \alpha'_{k-2}(n)$ at the first step of the algorithm, then both the running time of the algorithm and the number of edges in the resulting spanner G_T would increase by a factor of k, i.e., from $O(n\alpha_k(n))$ to $O(nk\alpha_k(n))$.
(2) In Sect. 2 we saw that $\alpha_{2\alpha(n)+2}(n) \leq 4$. Hence, we can compute in $O(n)$ time a 1-spanner for T with diameter at most $2\alpha(n) + 2$ and $O(n)$ edges.

The next lemma, which was essentially proved in [9, 26], bounds the size of the edge set E''.

LEMMA 3.8. *The edge set E'' contains at most $2n$ edges. Moreover, it can be computed in $O(n)$ time.*

Proof. Every edge of E'' is incident on exactly one cut vertex. Consider such an edge $(u, v) \in E''$, where $u \in CV_\ell$ and $v \in R(T) \setminus CV_\ell$. Then v belongs to some subtree T_i in $T \setminus CV_\ell$. We say that the edge (u, v) is *upstream* if u is the parent of the root $rt(T_i)$ of the subtree to which v belongs. Otherwise, the edge (u, v) is called *downstream.* (See Fig. 3 for an illustration.)

By definition, each vertex $v \in R \setminus CV_\ell$ is incident on at most one upstream edge. Hence, there are at most $|R \setminus CV_\ell| \leq |R| = n$ upstream edges in total.

The downstream edges are counted per cut vertex. Each cut vertex $u \in CV_\ell \setminus \{rt\}$ has one parent in T, denoted $\pi_T(u)$. If $\pi_T(u) \in CV_\ell$, then no downstream edge is incident on u. Otherwise, $\pi_T(u)$ belongs to some subtree $T_i \in T \setminus CV_\ell$. Each downstream edge that is incident on u belongs to a distinct required vertex in T_i. Hence, the first assertion of Lemma 3.7 implies that u is incident on at most ℓ downstream edges. By the second assertion of Lemma 3.7, $|CV_\ell| \le \lfloor \frac{n}{\ell+1} \rfloor$. Summing over all vertices in $CV_\ell \setminus \{rt\}$, we get a total of at most $\lfloor \frac{n}{\ell+1} \rfloor \ell \le n$ downstream edges. Hence, there are overall at most $2n$ edges in E''.

To verify that E'' can indeed be constructed within $O(n)$ time, we refer to Exercise 12.4 in [26]. ∎

Next, we prove Theorem 3.1 in the particular case of $k = 2$.

Lemma 3.9. *Let (T, rt) be a pruned tree with required-size $n \ge 0$. Algorithm $Tree1Spanner((T, rt), n, 2)$ computes in $O(n\alpha_2(n))$ time a 1-spanner $G_T = (V(T), E)$ for T with diameter at most 2 and at most $n\alpha_2(n)$ edges.*

Proof. We denote by $F_2(n)$ the maximum number of edges in the graph computed by Algorithm $Tree1Spanner((T, rt), n, 2)$, where T ranges over all pruned trees having required-size n. We next prove by induction on n that $F_2(n) \le n\alpha_2(n)$. Let T be a pruned tree with required-size n for which the edge set E that is computed by Algorithm $Tree1Spanner((T, rt), n, 2)$ has $F_2(n)$ edges. The case $n = 0$ is trivial. We henceforth assume that $n \ge 1$. By Lemma 3.5, any non-empty pruned tree is compact. Hence, $|V(T)| \le 2|R(T)| - 1 = 2n - 1$, and so $|E(T)| = |V(T)| - 1 \le 2n - 2$.
If $n \le 2$, then $F_2(n) = |E| = |E(T)| \le 2n - 2$. If $n = 1$, then $\alpha_2(n) = 0$, yielding $F_2(n) \le 2n - 2 = 0 = n\alpha_2(n)$. If $n = 2$, then $\alpha_2(n) = 1$, yielding $F_2(n) \le 2n - 2 = 2 = n\alpha_2(n)$.
Suppose next that $n = 3$. In this case the edge set E returned by the algorithm contains at most one more edge in addition to the edge set $E(T)$ of the input tree T. Hence, $F_2(n) = |E| \le |E(T)| + 1 \le 2n - 1 = 5$. Notice that $\alpha_2(3) = 2$, yielding $F_2(n) \le 5 \le n\alpha_2(n)$.
Induction Step: We assume the correctness of the statement for all smaller values of n, $n \ge 4$, and prove it for n. Notice that $\ell = \alpha'_0(n) = \alpha_0(n) = \lceil n/2 \rceil$. By the second assertion of Lemma 3.7, $|CV_\ell| \le \left\lfloor \frac{n}{\ell+1} \right\rfloor \le \left\lfloor \frac{n}{n/2+1} \right\rfloor = 1$. Thus CV_ℓ consists of a single vertex, denoted w.
Observe that for $k = 2$, the edge set E' is empty.
Since CV_ℓ consists of a single vertex w, we have $|E''| = |R(T) \setminus \{w\}| \le |R(T)| = n$.
Consider the edge set E_i, for some index $i \in [g]$. We have $|E_i| \le F_2(|R_i|)$. By the first assertion of Lemma 3.7, the required-size of each subtree in $T \setminus CV_\ell = T \setminus \{w\}$ is at most ℓ, and so $|R_i| \le \ell = \lceil n/2 \rceil < n$. Since the function α_2 is monotone non-decreasing, the induction hypothesis implies that $|E_i| \le |R_i|\alpha_2(\ell)$. Since $n \ge 4$, we have $\alpha_2(n) = 1 + \alpha_2(\alpha_0(n)) = 1 + \alpha_2(\ell)$. Also, notice that $\sum_{i=1}^{g} |R_i| \le |R| = n$. It follows that

$$\sum_{i=1}^{g} |E_i| \;\le\; \sum_{i=1}^{g} |R_i|\alpha_2(\ell) \;\le\; n(\alpha_2(n) - 1).$$

Altogether,

$$F_2(n) \;=\; |E| \;=\; |E'| + |E''| + \sum_{i=1}^{g} |E_i| \;\le\; n\alpha_2(n).$$

Next, we prove that G_T is a 1-spanner for T with diameter at most 2. The proof is, again, by induction on n. The case $n \le 3$ follows from Lemma 3.6 and Corollary 3.3.
Induction Step: We assume the correctness of the statement for all smaller values of n, $n \ge 4$, and prove it for n. We show that for an arbitrary pair u, v of required vertices, there is a T-monotone path in G_T that consists of at most two edges. Consider the single vertex w in CV_ℓ. If either u or v is equal to w, then u and v are connected by an edge of E'', and so this edge forms a T-monotone path between u and v. If u and v are in different subtrees of $T \setminus \{w\}$, then both edges (u, w) and (w, v) belong to G_T, and so u and v are connected by the path $P = (u, w, v)$ in G_T. Notice that the unique path $P_T(u, v)$ between u and v must traverse w, implying that P is T-monotone. Finally, if u and v belong to the same subtree T_i in $T \setminus \{w\}$, then by the induction hypothesis they are connected by a T_i-monotone path P_i that consists of at most two edges. However, P_i is also a path in G_T, and it is T-monotone.

Denote by $C_2(n)$ the worst-case running time of Algorithm $Tree1Spanner((T, rt), n, 2)$, where T ranges over all pruned trees with required-size n. We next show that $C_2(n) = O(n\alpha_2(n))$. Clearly, if $n \le 3$, then $C_2(n) = O(1)$. We may henceforth assume that $n \ge 4$. Computing the set CV_ℓ of cut vertices at the first step of the algorithm takes $O(n)$ time. Also, $E' = \emptyset$, and so the second step of the algorithm requires only $O(1)$ time. It takes $O(n)$ time to compute the subtrees $T_1, \ldots, T_g$ and the corresponding pruned subtrees $T'_1, \ldots, T'_g$ at the third and fifth steps of the algorithm, respectively. Recall that CV_ℓ consists of a single vertex w, and so one can compute the edge set $E'' = \{(w, v) \mid v \in R(T) \setminus \{w\}\}$ directly within $O(n)$ time. Finally, the time needed to compute the edge sets $E_1, E_2, \ldots, E_g$ at the fifth step of the algorithm is at most $\sum_{i=1}^{g} C_2(|R_i|)$. We obtain the recurrence $C_2(n) = O(n) + \sum_{i=1}^{g} C_2(|R_i|)$, where $|R_i| \le \ell = \lceil n/2 \rceil$, for each index $i \in [g]$, and $\sum_{i=1}^{g} |R_i| \le n$. Hence, as in the above argument for bounding $F_2(n)$, it can be shown that $C_2(n) = O(n\alpha_2(n))$. ∎

Next, we prove Theorem 3.1 in the particular case of $k = 3$.

LEMMA 3.10. *Let (T, rt) be a pruned tree with required-size $n \geq 0$. Algorithm $Tree1Spanner((T, rt), n, 3)$ computes in $O(n\alpha_3(n))$ time a 1-spanner $G_T = (V(T), E)$ for T with diameter at most 3 and $\frac{5}{2}n\alpha_3(n) + 2$ edges.*

Proof. We denote by $F_3(n)$ the maximum number of edges in the graph computed by Algorithm $Tree1Spanner((T, rt), n, 3)$, where T ranges over all pruned trees having required-size n. We next prove by induction on n that $F_3(n)$ is no greater than $\max\{2, \frac{5}{2}n\alpha_3(n)\}$, which provides the required result. Let T be a pruned tree with required-size n for which the edge set E that is computed by Algorithm $Tree1Spanner((T, rt), n, 3)$ has $F_3(n)$ edges. The case $n = 0$ is trivial. We henceforth assume that $n \geq 1$. Observe that $\max\{2, \frac{5}{2}n\alpha_3(n)\} = \frac{5}{2}n\alpha_3(n)$, for all $n \geq 3$. By Lemma 3.5, every non-empty pruned tree is compact. Hence, $|V(T)| \leq 2|R(T)| - 1 = 2n - 1$, and so $|E(T)| = |V(T)| - 1 \leq 2n - 2$. If $n \leq 3$, then $F_3(n) = |E| = |E(T)| \leq 2n - 2$. For $n \leq 2$, we have $F_3(n) \leq 2n - 2 \leq 2$. For $n = 3$, we have $\alpha_3(3) = 1$, and so $F_3(n) \leq 2n - 2 = 4 \leq \frac{5}{2}n\alpha_3(n)$. In both cases, we have $F_3(n) \leq \max\{2, \frac{5}{2}n\alpha_3(n)\}$.
Suppose next that $n = 4$. In this case the edge set E returned by the algorithm contains at most one more edge in addition to the edge set $E(T)$ of the input tree T. Hence, $F_3(n) = |E| \leq |E(T)| + 1 \leq 2n - 1 = 7$. Since $\alpha_3(4) = 1$, $F_3(n) \leq 7 \leq \frac{5}{2}n\alpha_3(n) = \max\{2, \frac{5}{2}n\alpha_3(n)\}$.
Induction Step: We assume the correctness of the statement for all smaller values of n, $n \geq 5$, and prove it for n. Notice that $\ell = \alpha'_1(n) = \alpha_1(n) = \lceil\sqrt{n}\rceil$. By the second assertion of Lemma 3.7, $|CV_\ell| \leq \left\lfloor \frac{n}{\ell+1} \right\rfloor \leq \sqrt{n}$.
Observe that for $k = 3$, the edge set E' consists of all $\binom{|CV_\ell|}{2}$ edges of the complete graph over CV_ℓ, and so $|E'| = \binom{|CV_\ell|}{2} \leq \binom{\sqrt{n}}{2} \leq \frac{n}{2}$.
By Lemma 3.8, we have $|E''| \leq 2n$.
Let I_1^-, I_2, and I_3^+ be the sets of all indices $i \in [g]$ for which $|R_i| \leq 1$, $|R_i| = 2$, and $|R_i| \geq 3$, respectively. Clearly, $I_1^- \cup I_2 \cup I_3^+ = [g]$. Observe that

$$\begin{aligned} n &\geq \sum_{i=1}^{g} |R_i| = \sum_{i \in I_1^-} |R_i| + \sum_{i \in I_2} |R_i| + \sum_{i \in I_3^+} |R_i| \\ &\geq 2|I_2| + \sum_{i \in I_3^+} |R_i|, \end{aligned}$$

implying that $\sum_{i \in I_3^+} |R_i| \leq n - 2|I_2|$. Consider the edge set E_i, for some index $i \in [g]$. We have $|E_i| \leq F_3(|R_i|)$. Observe that if $i \in I_1^-$, then $|E_i| = E(T'_i) = 0$, and if $i \in I_2$, then $|E_i| = E(T'_i) \leq 2$. Suppose next that $i \in I_3^+$. By the first assertion of Lemma 3.7, the required-size of each subtree in $T \setminus CV_\ell$ is at most ℓ, and so $3 \leq |R_i| \leq \ell = \lceil\sqrt{n}\rceil < n$. Since the function α_3 is monotone non-decreasing, the induction hypothesis implies that $|E_i| \leq \max\{2, \frac{5}{2}|R_i|\alpha_3(\ell)\} = \frac{5}{2}|R_i|\alpha_3(\ell)$. Since $n \geq 5$, we have $2 \leq \alpha_3(n) = 1 + \alpha_3(\alpha_1(n)) = 1 + \alpha_3(\ell)$. It follows that

$$\begin{aligned} \sum_{i=1}^{g} |E_i| &\leq 2|I_2| + \sum_{i \in I_3^+} |E_i| \\ &\leq 2|I_2| + \sum_{i \in I_3^+} \frac{5}{2}|R_i|(\alpha_3(n) - 1) \\ &\leq 2|I_2| + \frac{5}{2}(n - 2|I_2|)(\alpha_3(n) - 1) \\ &\leq \frac{5}{2}n(\alpha_3(n) - 1). \end{aligned}$$

Altogether,

$$\begin{aligned} F_3(n) &= |E| = |E'| + |E''| + \sum_{i=1}^{g} |E_i| \\ &\leq \frac{n}{2} + 2n + \frac{5}{2}n(\alpha_3(n) - 1) = \frac{5}{2}n\alpha_3(n). \end{aligned}$$

Next, we prove that G_T is a 1-spanner for T with diameter at most 3. The proof is, again, by induction on n. The case $n \leq 4$ follows from Lemma 3.6 and Corollary 3.3.
Induction Step: We assume the correctness of the statement for all smaller values of n, $n \geq 5$, and prove it for n. Observe that for $k = 3$, the edge set E' is equal to the edge set of the complete graph over CV_ℓ, and so there is an edge in G_T between any pair of vertices in CV_ℓ. Next, we show that for an arbitrary pair u, v of required vertices, there is a T-monotone path in G_T that consists of at most three edges.
Suppose first that $u, v \in T_i$, for some subtree T_i in $T \setminus CV_\ell$. The first assertion of Lemma 3.7 implies that the required-size $|R_i| = |R(T_i)|$ of T_i is at most $\ell = \alpha'_1(n) = \lceil\sqrt{n}\rceil < n$. By Lemma 3.1, $R(T'_i) = R(T_i)$. Hence, by the induction hypothesis for T'_i, the T'_i-monotone diameter of the graph $G_{T'_i} = (V(T'_i), E_i)$ is at most 3. Thus, u and v are connected in G_T by a T'_i-monotone path that consists of at most three edges. However, since T'_i is T_i-monotone preserving, this path is also T_i-monotone, and thus also T-monotone.
Otherwise, let w and w' be the first and last vertices of CV_ℓ on the path in T from u to v, respectively. Note that both w and w' belong to CV_ℓ. Hence, they are connected by the edge (w, w') in G_T, if $w \neq w'$. In addition, if $u \neq w$, then u belongs to some subtree $T_i \in T \setminus CV_\ell$. In this case, w is a border vertex of T_i, and so the edge (u, w) belongs to E'', and thus also to G_T. Similarly, if $v \neq w'$, the edge (w', v) belongs to G_T. By concatenating the edge (u, w), if $u \neq w$, the

edge (w, w'), if $w \neq w'$, and the edge (w', v), if $v \neq w'$, we get a T-monotone path in G_T between u and v that consists of at most three edges.

Denote by $C_3(n)$ the worst-case running time of Algorithm $Tree1Spanner((T, rt), n, 3)$, where T ranges over all pruned trees with required-size n. We next show that $C_3(n) = O(n\alpha_3(n))$. Clearly, if $n \leq 4$, then $C_3(n) = O(1)$. We may henceforth assume that $n \geq 5$. Computing the set CV_ℓ of cut vertices at the first step of the algorithm takes $O(n)$ time. Also, computing the edge set E' of the complete graph over CV_ℓ at the second step of the algorithm can be carried out in $O(|E'|) = O(n)$ time. It takes $O(n)$ time to compute the subtrees $T_1, \ldots, T_g$ and the corresponding pruned subtrees $T'_1, \ldots, T'_g$ at the third and fifth steps of the algorithm, respectively. By Lemma 3.8, computing the edge set E'' at the fourth step of the algorithm takes $O(n)$ time as well. Finally, the time needed to compute the edge sets $E_1, E_2, \ldots, E_g$ at the fifth step of the algorithm is at most $\sum_{i=1}^{g} C_3(|R_i|)$. We obtain the recurrence $C_3(n) = O(n) + \sum_{i=1}^{g} C_3(|R_i|)$, where $|R_i| \leq \ell = \lceil \sqrt{n} \rceil$, for each index $i \in [g]$, and $\sum_{i=1}^{g} |R_i| \leq n$. Hence, as in the above argument for bounding $F_3(n)$, it can be shown that $C_3(n) = O(n\alpha_3(n))$. ∎

We turn to prove Theorem 3.1 for a general k, $k \geq 2$.

The following lemma establishes an upper bound on the number of edges in the spanner G_T.

LEMMA 3.11. *Let $k \geq 2$ and $n \geq 0$ be two arbitrary integers, and denote by $F_k(n)$ the maximum number of edges in the graph computed by Algorithm $Tree1Spanner((T, rt), n, k)$, where T ranges over all pruned trees having required-size n. Then $F_k(n) \leq 2n\alpha'_k(n)$ if k is even, and $F_k(n) \leq 3n\alpha'_k(n) + 2$ if k is odd.*

Remark: By Lemma 2.4, for all $k, n \geq 0$, $\alpha'_k(n) \leq 2\alpha_k(n) + 4$. Hence, $F_k(n) = O(n\alpha_k(n))$.

Proof. We first give the proof for even values of k. The proof is by double induction on k and n. Let T be a pruned tree with required-size n for which the edge set E that is computed by algorithm $Tree1Spanner((T, rt), n, k)$ has $F_k(n)$ edges. The case $n = 0$ is trivial. Also, the case $k = 2$ follows from Lemma 3.9. We henceforth assume that $k \geq 4, n \geq 1$. By Lemma 3.5, every non-empty pruned tree is compact. Hence, $|V(T)| \leq 2|R(T)| - 1 = 2n - 1$, and so $|E(T)| = |V(T)| - 1 \leq 2n - 2$. The case $n = 1$ is now obvious. Suppose next that $2 \leq n \leq k + 1$. In this case $F_k(n) = |E| \leq |E(T)| + 1 \leq 2n - 1 \leq 2n\alpha'_k(n)$; the last inequality holds since $\alpha'_k(n) \geq \alpha_k(n) \geq 1$, for all $n \geq 2$.

Induction Step: We assume that for an arbitrary pair (k, n), $n \geq k + 2 \geq 6$, the statement holds for all pairs (k', n'), with either $k' < k$ or both $k' = k$ and $n' < n$, and prove it for the pair (k, n). The number of edges in E' is at most $F_{k-2}(|CV_\ell|)$. Since $n \geq 6$, it holds that $\ell = \alpha'_{k-2}(n) \geq \alpha_{k-2}(n) \geq 1$. By the second assertion of Lemma 3.7, $|CV_\ell| \leq \lfloor \frac{n}{\ell+1} \rfloor < n$. Since the function α'_{k-2} is monotone non-decreasing, $\alpha'_{k-2}(|CV_\ell|) \leq \alpha'_{k-2}(n) = \ell$. Therefore, by the induction hypothesis for the pair $(k - 2, |CV_\ell|)$, we have

$$\begin{aligned} |E'| &\leq F_{k-2}(|CV_\ell|) \leq 2|CV_\ell|\alpha'_{k-2}(|CV_\ell|) \\ &\leq 2\left\lfloor \frac{n}{\ell+1} \right\rfloor \ell \leq 2n. \end{aligned}$$

By Lemma 3.8, we have $|E''| \leq 2n$.

Consider the edge set E_i, for some index $i \in [g]$. We have $|E_i| \leq F_k(|R_i|)$. The first assertion of Lemma 3.7 and the second assertion of Lemma 2.3 imply that $|R_i| \leq \ell = \alpha'_{k-2}(n) < n$. Since the function α'_k is monotone non-decreasing, the induction hypothesis for the pair $(k, |R_i|)$ implies that $|E_i| \leq 2|R_i|\alpha'_k(\ell)$. Since $n \geq k+2$, we have $\alpha'_k(n) = 2 + \alpha'_k(\alpha'_{k-2}(n)) = 2 + \alpha'_k(\ell)$. Also, notice that $\sum_{i=1}^{g} |R_i| \leq |R| = n$. It follows that $\sum_{i=1}^{g} |E_i| \leq \sum_{i=1}^{g} 2|R_i|\alpha'_k(\ell) = \sum_{i=1}^{g} 2|R_i|(\alpha'_k(n) - 2) \leq 2n(\alpha'_k(n) - 2)$. Altogether,

$$\begin{aligned} F_k(n) &= |E| = |E'| + |E''| + \sum_{i=1}^{g} |E_i| \\ &\leq 2n + 2n + 2n(\alpha'_k(n) - 2) = 2n\alpha'_k(n). \end{aligned}$$

We next prove the lemma for odd values of k. The proof is, again, by double induction on k and n. Let T be a pruned tree with required-size n for which the edge set E that is computed by algorithm $Tree1Spanner((T, rt), n, k)$ has $F_k(n)$ edges. The case $n = 0$ is trivial. Also, the case $k = 3$ follows from Lemma 3.10. We henceforth assume that $k \geq 5, n \geq 1$. By Lemma 3.5, every non-empty pruned tree is compact. Hence, $|V(T)| \leq 2|R(T)| - 1 = 2n - 1$, and so $|E(T)| = |V(T)| - 1 \leq 2n - 2$. If $n \leq 2$, then $F_k(n) = |E| = |E(T)| \leq 2n-2 \leq 2 \leq 3n\alpha'_k(n)+2$. Suppose next that $3 \leq n \leq k + 1$. In this case $F_k(n) = |E| \leq |E(T)|+1 \leq 2n-1 \leq 3n\alpha'_k(n)+2$; the last inequality holds since $\alpha'_k(n) \geq \alpha_k(n) \geq 1$, for all $n \geq 3$.

Induction Step: We assume that for an arbitrary pair (k, n), $n \geq k + 2 \geq 7$, the statement holds for all pairs (k', n'), with either $k' < k$ or both $k' = k$ and $n' < n$, and prove it for the pair (k, n).

The number of edges in E' is at most $F_{k-2}(|CV_\ell|)$. Since $n \geq 7$, we have $\ell = \alpha'_{k-2}(n) \geq \alpha_{k-2}(n) \geq 1$. By the second assertion of Lemma 3.7, we have $|CV_\ell| \leq \lfloor \frac{n}{\ell+1} \rfloor < n$.

Since the function α'_{k-2} is monotone non-decreasing, $\alpha'_{k-2}(|CV_\ell|) \le \alpha'_{k-2}(n) = \ell$. Therefore, by the induction hypothesis for the pair $(k-2, |CV_\ell|)$,

$$\begin{aligned} |E'| &\le F_{k-2}(|CV_\ell|) \le 3|CV_\ell|\alpha'_{k-2}(|CV_\ell|) + 2 \\ &\le 3\left\lfloor \frac{n}{\ell+1} \right\rfloor \ell + 2 \le 3n + 2. \end{aligned}$$

By Lemma 3.8, we have $|E''| \le 2n$.
Let I_1^- (respectively, I_2^+) be the set of all indices i, such that $i \in [g]$ and $|R_i| \le 1$ (resp., $|R_i| \ge 2$). Clearly, $I_1^- \cup I_2^+ = [g]$. Observe that

$$n = |R| \ge \sum_{i=1}^{g} |R_i| = \sum_{i \in I_1^-} |R_i| + \sum_{i \in I_2^+} |R_i| \ge 2|I_2^+|.$$

Consider the edge set E_i, for some index $i \in [g]$. We have $|E_i| \le F_k(|R_i|)$. Observe that if $i \in I_1^-$, we have $|E_i| = |E(T'_i)| = 0$. Suppose next that $i \in I_2^+$. The first assertion of Lemma 3.7 and the second assertion of Lemma 2.3 imply that $|R_i| \le \ell = \alpha'_{k-2}(n) < n$. Since the function α'_k is monotone non-decreasing, the induction hypothesis for the pair $(k, |R_i|)$ implies that $|E_i| \le 3|R_i|\alpha'_k(\ell) + 2$. Since $n \ge k+2$, we have $\alpha'_k(n) = 2 + \alpha'_k(\alpha'_{k-2}(n)) = 2 + \alpha'_k(\ell)$. It follows that

$$\begin{aligned} \sum_{i=1}^{g} |E_i| &= \sum_{i \in I_2^+} |E_i| \le \sum_{i \in I_2^+} (3|R_i|\alpha'_k(\ell) + 2) \\ &= \sum_{i \in I_2^+} 3|R_i|(\alpha'_k(n) - 2) + 2|I_2^+| \\ &\le 3n(\alpha'_k(n) - 2) + n. \end{aligned}$$

Altogether,

$$\begin{aligned} F_k(n) &= |E| = |E'| + |E''| + \sum_{i=1}^{g} |E_i| \\ &\le (3n+2) + 2n + 3n(\alpha'_k(n) - 2) + n \\ &= 3n\alpha'_k(n) + 2. \quad \blacksquare \end{aligned}$$

The next lemma demonstrates that G_T is a 1-spanner for T with diameter at most k.

Lemma 3.12. *Let $k \ge 2$ and $n \ge 0$ be two arbitrary integers. For any pruned tree (T, rt) with required-size n, the T-monotone diameter $\Lambda(G_T)$ of the graph $G_T = (V(T), E)$ that is computed by Algorithm $Tree1Spanner((T, rt), n, k)$ is at most k.*

Proof. The proof is by double induction on k and n. The cases $k = 2$ and $k = 3$ follow from Lemmas 3.9 and 3.10, respectively. We henceforth assume that $k \ge 4$.
For $0 \le n \le k+1$, the correctness of the statement follows from Lemma 3.6 and Corollary 3.3.
Induction Step: We assume that for an arbitrary pair (k, n), $n \ge k+2 \ge 6$, the statement holds for all pairs (k', n'), with either $k' < k$ or both $k' = k$ and $n' < n$, and prove it for the pair (k, n). By Lemma 3.1, the tree τ' that is constructed at the second step of the algorithm satisfies $R(\tau') = R(\tau) = CV_\ell$. By the induction hypothesis for the pair $(k-2, |CV_\ell|)$, the τ'-monotone diameter of the graph $G_{\tau'} = (V(\tau'), E')$ that is computed at the second step of the algorithm is at most $k-2$. Since τ' is τ-monotone preserving and τ is a copy of T, it follows that there is a T-monotone path in G_T between any two vertices of CV_ℓ that consists of at most $k-2$ edges. Next, we show that for an arbitrary pair u, v of required vertices, there is a T-monotone path in G_T that consists of at most k edges.

Suppose first that $u, v \in T_i$, for some subtree T_i in $T \setminus CV_\ell$. The first assertion of Lemma 3.7 and the second assertion of Lemma 2.3 imply that the required-size $|R_i| = |R(T_i)|$ of T_i is at most $\ell = \alpha'_{k-2}(n) < n$. By Lemma 3.1, the tree T'_i that is computed at the fifth step of the algorithm satisfies $R(T'_i) = R(T_i)$. Hence, by the induction hypothesis for the pair $(k, |R_i|)$, the T'_i-monotone diameter of the graph $G_{T'_i} = (V(T'_i), E_i)$ that is computed at the fifth step of the algorithm is at most k. It follows that u and v are connected in G_T by a T'_i-monotone path that consists of at most k edges. However, since T'_i is T_i-monotone preserving, this path is also T_i-monotone, and thus also T-monotone.

Otherwise, let w and w' be the first and last vertices of CV_ℓ on the path in T from u to v, respectively. Since both w and w' belong to CV_ℓ, the graph G_T contains a T-monotone path P between w and w' that consists of at most $k-2$ edges. In addition, if $u \ne w$, then u belongs to some subtree $T_i \in T \setminus CV_\ell$. In this case, w is a border vertex of T_i, and so the edge (u, w) belongs to E'', and thus also to G_T. Similarly, if $v \ne w'$, the edge (w', v) belongs to G_T. By concatenating the edge (u, w), if $u \ne w$, the path P, and the edge (w', v), if $v \ne w'$, we get a T-monotone path in G_T between u and v that consists of at most k edges. $\blacksquare$

Finally, we bound the running time of the algorithm $Tree1Spanner((T, rt), n, k)$.

Lemma 3.13. *Let $k \ge 2$ and $n \ge 0$ be two arbitrary integers, and denote by $C_k(n)$ the worst-case running time of Algorithm $Tree1Spanner((T, rt), n, k)$, where T ranges over all pruned trees with required-size n. Then $C_k(n) = O(n\alpha_k(n))$.*

Proof. Clearly, if $n \le k+1$, then $C_k(n) = O(n)$. We may henceforth assume that $n \ge k+2$.
We remark that one can compute the values of the

function $\alpha'_k = \alpha'_k(n)$ in $O(n)$ time, for all $k \geq 2$ and $n \geq k+2$. These values can be computed similarly to the way the values of the function $\alpha_k = \alpha_k(n)$ are computed in [23]. (See also Exercise 12.7 in [26]; further details on this technical argument are omitted.) In particular, computing the value of $\alpha'_{k-2}(n)$ with which ℓ is assigned at the first step of the algorithm can be carried out in $O(n)$ time. Also, an additional time of $O(n)$ suffices to compute the set CV_ℓ of cut vertices at the first step of the algorithm. The computation of the edge set E' at the second step of the algorithm starts by computing a copy τ of T, which can be carried out in $O(n)$ time. Another $O(n)$ time is required to go over the vertices of τ and designate the vertices of CV_ℓ as the required vertices of τ. Computing the pruning τ' of τ also requires $O(n)$ time. Finally, the recursive call $Tree1Spanner((\tau', rt(\tau')), |CV_\ell|, k-2)$ requires at most $C_{k-2}(|CV_\ell|)$ time. Overall, the time needed to compute E' is bounded above by $O(n) + C_{k-2}(|CV_\ell|)$. An additional amount of $O(n)$ time is needed to compute the subtrees $T_1, \ldots, T_g$ and the corresponding pruned subtrees $T'_1, \ldots, T'_g$ at the third and fifth steps of the algorithm, respectively. By Lemma 3.8, computing the edge set E'' at the fourth step of the algorithm takes another $O(n)$ time. Finally, the time needed to compute the edge sets $E_1, E_2, \ldots, E_g$ at the fifth step of the algorithm is at most $\sum_{i=1}^{g} C_k(|R_i|)$. We obtain the recurrence $C_k(n) = O(n) + C_{k-2}(|CV_\ell|) + \sum_{i=1}^{g} C_k(|R_i|)$, where $|CV_\ell| \leq \left\lfloor \frac{n}{\ell+1} \right\rfloor$, $|R_i| \leq \ell = \alpha'_{k-2}(n)$, for each index $i \in [g]$, and $\sum_{i=1}^{g} |R_i| \leq n$. Hence, as in the proof of Lemma 3.11, it can be shown that $C_k(n) = O(n\alpha'_k(n)) = O(n\alpha_k(n))$. ∎

Lemmas 3.11, 3.12 and 3.13 imply Theorem 3.1.

4 Sparse Euclidean Spanners with Bounded Diameter

In this section we plug the 1-spanners for tree metrics from Sect. 3 on top of the dumbbell trees of [5] to obtain our construction of Euclidean spanners.

Theorem 4.1. *("Dumbbell Theorem", Theorem 2 in [5]) Let $d \geq 2$ be an integer constant. Given a set S of n points in $\mathbb{R}^d$ and a parameter $\epsilon > 0$, a forest $\mathcal{F}$ consisting of $O(\frac{\log(1/\epsilon)}{\epsilon^d})$ rooted trees of size $O(n)$ each can be built in $O(\frac{\log(1/\epsilon)}{\epsilon^d}(n \log n))$ time, having the following properties:*

1. *For each tree in $\mathcal{F}$, there is a 1-1 correspondence between the leaves of this tree and the points of S.*
2. *Each internal vertex in the tree has a unique representative point, which can be selected arbitrarily from the points in any of its descendant leaves.*
3. *For any two points $u, v \in S$, there is a tree in $\mathcal{F}$, so that the path formed by walking from representative to representative along the unique path in that tree between u and v, is a $(1+\epsilon)$-spanner path.*

Let S be a set of n points in $\mathbb{R}^d$, let $\mathcal{F}$ be the forest of dumbbell trees given by the Dumbbell Theorem, and let $k \geq 2$ be an arbitrary integer. For each dumbbell tree $T \in \mathcal{F}$, let G_T be the 1-spanner for T that is guaranteed by Theorem 3.1, having diameter at most k and $O(n\alpha_k(n))$ edges. Our construction of Euclidean spanners is defined to be the geometric graph $\mathcal{G}_k(n)$ implied by the collection of all the graphs G_T, $T \in \mathcal{F}$.

Since each graph G_T has only $O(n\alpha_k(n))$ edges, the collection of $O(\frac{\log(1/\epsilon)}{\epsilon^d})$ such graphs will have at most $O(\frac{\log(1/\epsilon)}{\epsilon^d}(n\alpha_k(n)))$ edges.

By the Dumbbell Theorem, the forest $\mathcal{F}$ of dumbbell trees can be built in $O(\frac{\log(1/\epsilon)}{\epsilon^d}(n \log n))$ time. By Theorem 3.1, we can compute each of the graphs G_T within time $O(n\alpha_k(n)) = O(n \log n)$. Since there are $O(\frac{\log(1/\epsilon)}{\epsilon^d})$ such graphs, we get that the overall time needed to compute our construction $\mathcal{G}_k(n)$ of Euclidean spanners is $O(\frac{\log(1/\epsilon)}{\epsilon^d}(n \log n))$.

Finally, we show that $\mathcal{G}_k(n)$ is a $(1+\epsilon)$-spanner for S with diameter at most k. Consider an arbitrary pair of points $u, v \in S$. By the Dumbbell Theorem, there is a dumbbell tree $T \in \mathcal{F}$, so that the geometric path $\mathcal{P}_T(u, v)$ implied by the unique path $P_T(u, v)$ between u and v in T is a $(1+\epsilon)$-spanner path. Theorem 3.1 implies that there is a 1-spanner path P for T between u and v in G_T that consists of at most k edges. By the triangle inequality, the weight of the corresponding geometric path $\mathcal{P}$ in $\mathcal{G}_k(n)$ is no greater than the weight of $\mathcal{P}_T(u, v)$. Hence, $\mathcal{P}$ is a $(1+\epsilon)$-spanner path for u and v that consists of at most k edges.

We derive the following corollary, which settles the open question of [5, 26] in the affirmative.

Corollary 4.1. *Let $d \geq 2$ be an integer constant. For any set of n points in $\mathbb{R}^d$, any number $\epsilon > 0$ and an integer $k \geq 2$, we can compute in $O(\frac{\log(1/\epsilon)}{\epsilon^d}(n \log n))$ time a $(1+\epsilon)$-spanner with diameter at most k and $O(\frac{\log(1/\epsilon)}{\epsilon^d}(n\alpha_k(n)))$ edges.*

5 Lower Bounds for Euclidean Steiner Spanners

In this section we extend the lower bound of Chan and Gupta [11] to Euclidean Steiner spanners.

Theorem 5.1. *Let X be a set of n points on the x-axis, and let $H = (V, E)$, $X \subseteq V$, be a Euclidean Steiner t-spanner for X, with $t \geq 1$, having diameter Λ and m edges. Then H can be transformed into a Euclidean t-*

spanner $H' = (X, E')$ with diameter at most Λ and at most $4m$ edges.

Proof. For every point $p \in \mathbb{R}^d$, denote by $p(x)$ its projection onto the x-axis. Let $S = V \setminus X$ be the set of Steiner points of H, and let $\tilde{S}$ be the set of all projections of the points in S onto the x-axis, i.e., $\tilde{S} = \{v(x) \mid v \in S\}$. Also, define $\tilde{V} = X \cup \tilde{S}$, and let $\tilde{H} = (\tilde{V}, \tilde{E})$ be the graph obtained from H by replacing each edge $e = (u, v)$ with its projection $\tilde{e} = (u(x), v(x))$ onto the x-axis. Clearly, $\tilde{H}$ is a spanning subgraph over a superset $\tilde{V}$ of X of points that lie on the x-axis, having $|\tilde{E}| = |E| = m$ edges. Also, it is easy to see that for every pair u, v of points in V, and every path $P = (u = v_0, v_1, \ldots, v = v_i)$ in H between u and v, the weight $w(\tilde{P})$ of the corresponding path $\tilde{P} = (u(x) = v_0(x), v_1(x), \ldots, v(x) = v_i(x))$ in $\tilde{H}$ is no greater than the weight $w(P)$ of P. Hence, $\tilde{H}$ is a Euclidean Steiner t-spanner for X over a superset $\tilde{V}$ of X of points that lie on the x-axis, having diameter at most Λ and m edges.

For every point $v \in \tilde{V}$, denote by v_L (respectively, v_R) the point closest to v among all points in X that are located left (resp., right) to v on the x-axis, including v itself. If $v \in X$, then $v_L = v_R = v$. If there is no point in X to the left (respectively, right) of v, then we write $v_L = NULL$ (resp., $v_R = NULL$). Let $\hat{H}$ be the graph obtained from $\tilde{H}$ by replacing each edge $(u, v) \in \tilde{E}$ with the four edges (u_L, v_L), (u_L, v_R), (u_R, v_L), and (u_R, v_R). Notice that the resulting graph $\hat{H}$ may contain multiple copies of the same edge as well as self loops, and so $\hat{H}$ is, in fact, a multigraph. In addition, $\hat{H}$ may contain edges with one or two NULL endpoints. Next, we transform $\hat{H}$ into a simple graph H' by removing from it all the multiple edges, self loops, and edges with either one or two NULL endpoints. It is easy to see that no edge in the resulting graph H' is incident on a Steiner point. Moreover, H' contains at most $4m$ edges. To complete the proof of Theorem 5.1, we employ the following lemma. (In what follows we denote the number of edges in a path P by $|P|$.)

LEMMA 5.1. *Let u, v be an arbitrary pair of distinct points in $\tilde{V}$, let u' be either u_L or u_R, and let v' be either v_L or v_R, with $u', v' \neq NULL$. Then for any path $\tilde{P}$ between u and v in $\tilde{H}$, there exists a path P' between u' and v' in H', such that $|P'| \leq |\tilde{P}|$ and $w(P') \leq w(\tilde{P}) + \|u' - u\| + \|v' - v\|$.*

Proof. The proof is by induction on the number of edges $q = |\tilde{P}|$ in the path $\tilde{P}$.
Basis: $q = |\tilde{P}| = 1$. In this case $\tilde{P} = (u, v)$. The proof is immediate if $u' = v'$, and so we may assume that $u' \neq v'$. By construction, H' contains the edge (u', v'). Set $P' = (u', v')$. Clearly, $|P'| = |\tilde{P}| = 1$. Also, by the triangle inequality,

$$\begin{aligned} w(P') &= \|u' - v'\| \;\leq\; \|u - v\| + \|u' - u\| + \|v' - v\| \\ &= w(\tilde{P}) + \|u' - u\| + \|v' - v\|. \end{aligned}$$

Induction Step: We assume the correctness of the statement for all smaller values of q, $q \geq 2$, and prove it for q. Consider the second point w on the path $\tilde{P} = (u, w, \ldots, v)$ between u and v in $\tilde{H}$. Observe that either w_L or w_R is located on the line segment between u' and w. (In the case $u' = w$, we have $w_L = w_R = u' = w$.) Denote this vertex by w', and note that it is possible to have $u' = w'$, e.g., if $u' = w$. Since (u, w) is an edge in $\tilde{P} \in \tilde{H}$, it holds by construction that (u', w') is an edge in H'. Consider the sub-path $\tilde{P}_{w,v}$ of $\tilde{P}$ between w and v, i.e., the path obtained by removing the first edge (u, w) from $\tilde{P}$. It consists of $q - 1$ edges, and so $|\tilde{P}_{w,v}| = q - 1$. Hence, by the induction hypothesis, there exists a path $P'_{w',v'}$ between w' and v' in H', such that $|P'_{w',v'}| \leq |\tilde{P}_{w,v}| = q - 1$ and $w(P'_{w',v'}) \leq w(\tilde{P}_{w,v}) + \|w' - w\| + \|v' - v\|$. Since w' is located on the line segment between u' and w, it follows that $\|u' - w'\| + \|w' - w\| = \|u' - w\|$. By the triangle inequality, $\|u' - w\| \leq \|u' - u\| + \|u - w\|$. Let P' be the path obtained by concatenating the edge (u', w') with the path $P'_{w',v'}$, i.e., $P' = (u', w') \circ P'_{w',v'}$. Notice that P' is a path in H' between u' and v', and $|P'| = 1 + |P'_{w',v'}| \leq q$. Also, we have

$$\begin{aligned} w(P') &= \|u' - w'\| + w(P'_{w',v'}) \\ &\leq \|u' - w'\| + w(\tilde{P}_{w,v}) + \|w' - w\| + \|v' - v\| \\ &= \|u' - w\| + w(\tilde{P}_{w,v}) + \|v' - v\| \\ &\leq \|u' - u\| + \|u - w\| + w(\tilde{P}_{w,v}) + \|v' - v\| \\ &= w(\tilde{P}) + \|u' - u\| + \|v' - v\|. \qquad \blacksquare \end{aligned}$$

Lemma 5.1 implies that for any two points in X, there is a t-spanner path in H' that consists of at most Λ edges. Thus H' is a Euclidean t-spanner for X with diameter at most Λ and at most $4m$ edges. $\blacksquare$

Chan and Gupta [11] proved that for any $\epsilon > 0$, there exists a set S_ϵ of n points on the x-axis, where n is an arbitrary power of two, for which any Euclidean $(1 + \epsilon)$-spanner with at most m edges has diameter at least $\Omega(\alpha(m, n))$. Theorem 5.1 enables us to extend the lower bound of [11] to Euclidean Steiner spanners.

COROLLARY 5.1. *For any $\epsilon > 0$, there exists a set of n points on the x-axis, for which any Euclidean (possibly Steiner) $(1 + \epsilon)$-spanner with at most m edges has diameter at least $\Omega(\alpha(m, n))$.*

Proof. The statement is trivial if $\alpha(m,n) = O(1)$. We henceforth assume that $\alpha(m,n)$ is super-constant.

Let S_ϵ be the aforementioned set of n points for which the lower bound of [11] holds, and suppose for contradiction that there exists a Euclidean Steiner $(1+\epsilon)$-spanner H for S_ϵ with at most m edges and diameter $\Lambda = o(\alpha(m,n))$. By Theorem 5.1, we can transform H into a Euclidean $(1+\epsilon)$-spanner H' for S_ϵ, having diameter $\Lambda' \leq \Lambda = o(\alpha(m,n))$ and at most $4m$ edges. However, the lower bound of [11] implies that the diameter Λ' of H' is at least $\Omega(\alpha(4m,n))$. Using the observation that $\alpha(4m,n) \geq \alpha(m,n) - 4$, for all $m \geq n$, we conclude that $\Lambda' = \Omega(\alpha(m,n) - 4) = \Omega(\alpha(m,n))$, yielding a contradiction. ∎

Acknowledgments. The author is grateful to Michael Elkin and Michiel Smid for helpful discussions.

References

[1] I. Abraham and D. Malkhi. Compact routing on Euclidean metrics. In *Proc. of 23rd PODC*, pages 141–149, 2004.

[2] P. K. Agarwal, Y. Wang, and P. Yin. Lower bound for sparse Euclidean spanners. In *Proc. of 16th SODA*, pages 670–671, 2005.

[3] N. Alon and B. Schieber. Optimal preprocessing for answering on-line product queries. *Manuscript*, 1987.

[4] I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993.

[5] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. H. M. Smid. Euclidean spanners: short, thin, and lanky. In *Proc. of 27th STOC*, pages 489–498, 1995.

[6] S. Arya, D. M. Mount, and M. H. M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *Proc. of 35th FOCS*, pages 703–712, 1994.

[7] S. Arya and M. H. M. Smid. Efficient construction of a bounded degree spanner with low weight. *Algorithmica*, 17(1):33–54, 1997.

[8] A. Bhattacharyya, E. Grigorescu, K. Jung, S. Raskhodnikova, and D. P. Woodruff. Transitive-closure spanners. In *Proc. of 20th SODA*, pages 932–941, 2009.

[9] H. L. Bodlaender, G. Tel, and N. Santoro. Trade-offs in non-reversing diameter. *Nord. J. Comput.*, 1(1):111–134, 1994.

[10] P. B. Callahan and S. R. Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proc. of 4th SODA*, pages 291–300, 1993.

[11] H. T.-H. Chan and A. Gupta. Small hop-diameter sparse spanners for doubling metrics. In *Proc. of 17th SODA*, pages 70–78, 2006.

[12] B. Chazelle. Computing on a free tree via complexity-preserving mappings. *Algorithmica*, 2:337–361, 1987.

[13] B. Chazelle and B. Rosenberg. The complexity of computing partial sums off-line. *Int. J. Comput. Geom. Appl.*, 1:33–45, 1991.

[14] L. P. Chew. There is a planar graph almost as good as the complete graph. In *Proc. of 2nd SoCG*, pages 169–177, 1986.

[15] G. Das and G. Narasimhan. A fast algorithm for constructing sparse Euclidean spanners. In *Proc. of 10th SoCG*, pages 132–139, 1994.

[16] G. Das, G. Narasimhan, and J. S. Salowe. A new way to weigh malnourished Euclidean graphs. In *Proc. of 6th SODA*, pages 215–222, 1995.

[17] Y. Dinitz, M. Elkin, and S. Solomon. Shallow-low-light trees, and tight lower bounds for Euclidean spanners. In *Proc. of 49th FOCS*, pages 519–528, 2008.

[18] J. Gudmundsson, C. Levcopoulos, G. Narasimhan, and M. H. M. Smid. Approximate distance oracles for geometric graphs. In *Proc. of 13th SODA*, pages 828–837, 2002.

[19] J. Gudmundsson, C. Levcopoulos, G. Narasimhan, and M. H. M. Smid. Approximate distance oracles for geometric spanners. *ACM Transactions on Algorithms*, 4(1), 2008.

[20] J. Gudmundsson, G. Narasimhan, and M. H. M. Smid. Fast pruning of geometric spanners. In *Proc. of 22nd STACS*, pages 508–520, 2005.

[21] Y. Hassin and D. Peleg. Sparse communication networks and efficient routing in the plane. In *Proc. of 19th PODC*, pages 41–50, 2000.

[22] J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry*, 7:13–28, 1992.

[23] J. A. La Poutré. New techniques for the union-find problems. In *Proc. of 1st SODA*, pages 54–63, 1990.

[24] C. Levcopoulos, G. Narasimhan, and M. H. M. Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proc. of 30th STOC*, pages 186–195, 1998.

[25] Y. Mansour and D. Peleg. An approximation algorithm for min-cost network design. *DIMACS Series in Discr. Math and TCS*, 53:97–106, 2000.

[26] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

[27] M. Pătraşcu and E. D. Demaine. Tight bounds for the partial-sums problem. In *Proc. of 15th SODA*, pages 20–29, 2004.

[28] S. Rao and W. D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proc. of 30th STOC*, pages 540–550, 1998.

[29] M. H. M. Smid. Private communication.

[30] M. H. M. Smid. Progress on open problems mentioned in the book Geometric Spanner Networks. Available via http://people.scs.carleton.ca/∼michiel/SpannerBook/openproblems.html.

[31] S. Solomon and M. Elkin. Balancing degree, diameter and weight in Euclidean spanners. In *Proc. of 18th ESA*, Part 1, pages 48–59, 2010.

[32] R. E. Tarjan. Efficiency of a good but not linear set

union algorithm. *J. ACM*, 22(2):215–225, 1975.
[33] R. E. Tarjan. Applications of path compression on balanced trees. *J. ACM*, 26(4):690–715, 1979.
[34] M. Thorup. Parallel shortcutting of rooted trees. *J. Algorithms*, 23(1):139–159, 1997.
[35] A. C. Yao. Space-time tradeoff for answering range queries. In *Proc. of 14th STOC*, pages 128–136, 1982.

A Proof of Lemma 2.3

This section is devoted to the proof of Lemma 2.3.

We start with proving the following lemma.

LEMMA A.1. *(1) The function $\alpha'_2 = \alpha'_2(n)$ is monotone non-decreasing with n. (2) For all $n \geq 1$, $\alpha'_2(n) \leq n-1$. Moreover, if $n \geq 6$, then $\alpha'_2(n) \leq n-2$. (3) For all $n > 10$, $\alpha'_2(n) \leq \alpha'_0(n)$. (4) For all $n \geq 0$, $\alpha'_4(n) \leq \alpha'_2(n)$.*

Proof. We first prove that $\alpha'_2 = \alpha'_2(n)$ is monotone non-decreasing with n. Specifically, we show that for all $n \geq 0$: $\alpha'_2(m) \leq \alpha'_2(n)$, for any $m \leq n$. The proof is by induction on n. The basis $n \leq 3$ can be easily verified.
Induction Step: We assume the correctness of the statement for all smaller values of n, $n \geq 4$, and prove it for n. By definition, $\alpha'_2(n) = 2 + \alpha'_2(\alpha'_0(n)) = 2 + \alpha'_2(\lceil n/2 \rceil)$. It is easy to see that for $m \leq 3$, $\alpha'_2(m) \leq 2$, and so $\alpha'_2(m) \leq 2 \leq 2+\alpha'_2(\lceil n/2 \rceil) = \alpha'_2(n)$. We henceforth assume that $4 \leq m \leq n$. Thus, by definition, $\alpha'_2(m) = 2 + \alpha'_2(\alpha'_0(m)) = 2 + \alpha'_2(\lceil m/2 \rceil)$. Since $4 \leq m \leq n$, we have $\lceil m/2 \rceil \leq \lceil n/2 \rceil < n$. By the induction hypothesis for $\lceil n/2 \rceil$, it follows that $\alpha'_2(\lceil m/2 \rceil) \leq \alpha'_2(\lceil n/2 \rceil)$. Consequently,

$$\begin{aligned} \alpha'_2(m) &= 2 + \alpha'_2(\lceil m/2 \rceil) \\ &\leq 2 + \alpha'_2(\lceil n/2 \rceil) \ = \ \alpha'_2(n). \end{aligned}$$

Next, we prove the second assertion. It is easy to see that $\alpha'_2(n) = n - 1$, for all $1 \leq n \leq 5$. We prove by induction on n that for all $n \geq 6$, it holds that $\alpha'_2(n) \leq n - 2$. The basis $n = 6$ can be easily verified.
Induction Step: We assume the correctness of the statement for all smaller values of n, $n \geq 7$, and prove it for n. By definition, $\alpha'_2(n) = 2 + \alpha'_2(\alpha'_0(n)) = 2 + \alpha'_2(\lceil n/2 \rceil)$. Since $n \geq 7$, we have $4 \leq \lceil n/2 \rceil < n$. If $\lceil n/2 \rceil \leq 5$, then $\alpha'_2(\lceil n/2 \rceil) \leq \lceil n/2 \rceil - 1$. Otherwise, $\lceil n/2 \rceil \geq 6$, and by the induction hypothesis for $\lceil n/2 \rceil$, $\alpha'_2(\lceil n/2 \rceil) \leq \lceil n/2 \rceil - 2$. In any case, it holds that $\alpha'_2(\lceil n/2 \rceil) \leq \lceil n/2 \rceil - 1$. Consequently,

$$\alpha'_2(n) \ = \ 2 + \alpha'_2(\lceil n/2 \rceil) \ \leq \ 1 + \lceil n/2 \rceil \ \leq \ n - 2.$$

(The last inequality holds for all $n \geq 6$.)

To prove the third assertion, consider an arbitrary integer $n \geq 11$. By definition, $\alpha'_2(n) = 2 + \alpha'_2(\alpha'_0(n)) = 2 + \alpha'_2(\lceil n/2 \rceil)$. Since $\lceil n/2 \rceil \geq 6$, the second assertion of this lemma yields $\alpha'_2(\lceil n/2 \rceil) \leq \lceil n/2 \rceil - 2$, and so

$$\alpha'_2(n) \ = \ 2 + \alpha'_2(\lceil n/2 \rceil) \ \leq \ \lceil n/2 \rceil \ = \ \alpha'_0(n).$$

The proof of the fourth assertion is by induction on n. For all $0 \leq n \leq 10$ the statement can be verified by brute force.
Induction Step: We assume the correctness of the statement for all smaller values of n, $n \geq 11$, and prove it for n. By definition, for all $n \geq 11$, $\alpha'_4(n) = 2 + \alpha'_4(\alpha'_2(n))$ and $\alpha'_2(n) = 2 + \alpha'_2(\alpha'_0(n))$. By the third assertion of this lemma, we know that $\alpha'_2(n) \leq \alpha'_0(n)$. Hence, by the first assertion of this lemma, $\alpha'_2(\alpha'_0(n)) \geq \alpha'_2(\alpha'_2(n))$. The second assertion of this lemma implies that $\alpha'_2(n) < n$, and so by the induction hypothesis for $\alpha'_2(n)$, we have $\alpha'_4(\alpha'_2(n)) \leq \alpha'_2(\alpha'_2(n))$. Altogether,

$$\begin{aligned} \alpha'_4(n) &= 2 + \alpha'_4(\alpha'_2(n)) \ \leq \ 2 + \alpha'_2(\alpha'_2(n)) \\ &\leq 2 + \alpha'_2(\alpha'_0(n)) \ = \ \alpha'_2(n). \end{aligned}$$

∎

We now turn to the proof of Lemma 2.3.
The three assertions of the lemma are proved by double induction on k and n. We restrict the attention to even values of k. The argument for odd values of k is similar, and is thus omitted.
For technical convenience, we will prove the first assertion of the lemma in the sequel by showing that $\alpha'_k(m) \leq \alpha'_k(n)$, for an arbitrary integer $m \leq n$.
The case $k = 2$ follows from Lemma A.1.
Suppose next that $n \leq k + 1$. By definition, $\alpha'_k(n) = \alpha_k(n)$, for any $n \leq k + 1$, and so the three assertions of the lemma follow from Lemma 2.1.
Induction Step: We assume that for an arbitrary pair (k, n), $n \geq k + 2 \geq 6$, the three assertions of the lemma hold for all pairs (k', n'), with either $k' < k$ or both $k' = k$ and $n' < n$, and prove it for the pair (k, n).

Consider an arbitrary integer $m \leq n$. We first show that $\alpha'_k(m) \leq \alpha'_k(n)$, thus proving the first assertion.
If $m \leq k + 1$, then $\alpha'_k(m) = \alpha_k(m)$. Clearly, $\alpha_k(n) \leq \alpha'_k(n)$. By the first assertion of Lemma 2.1, $\alpha_k(m) \leq \alpha_k(n)$, and so

$$\alpha'_k(m) \ = \ \alpha_k(m) \ \leq \ \alpha_k(n) \ \leq \ \alpha'_k(n).$$

Otherwise, we have $k+2 \leq m \leq n$. Hence, by definition, $\alpha'_k(n) = 2+\alpha'_k(\alpha'_{k-2}(n))$ and $\alpha'_k(m) = 2+\alpha'_k(\alpha'_{k-2}(m))$. By the first and second assertions of the induction hypothesis for the pair $(k - 2, n)$, we have $\alpha'_{k-2}(m) \leq \alpha'_{k-2}(n)$ and $\alpha'_{k-2}(n) < n$, respectively. Hence, the first assertion of the induction hypothesis for the pair $(k, \alpha'_{k-2}(n))$ implies that $\alpha'_k(\alpha'_{k-2}(m)) \leq \alpha'_k(\alpha'_{k-2}(n))$. Altogether,

$$\begin{aligned} \alpha'_k(m) &= 2 + \alpha'_k(\alpha'_{k-2}(m)) \\ &\leq 2 + \alpha'_k(\alpha'_{k-2}(n)) \ = \ \alpha'_k(n). \end{aligned}$$

The second and third assertions of the induction hypothesis for the pair $(k-2, n)$ imply that

$$\text{(A.1)} \qquad \alpha'_k(n) \ \leq \ \alpha'_{k-2}(n) \ < \ n,$$

thus proving the second assertion of the lemma.

We next prove the third assertion. Suppose first that $k+2 \leq n \leq k+3$. In this case $\alpha'_{k+2}(n) = \alpha_{k+2}(n)$. Also, we have $\alpha'_k(n) \geq \alpha_k(n)$. By the third assertion of Lemma 2.1, $\alpha_{k+2}(n) \leq \alpha_k(n)$, yielding

$$\alpha'_{k+2}(n) \ = \ \alpha_{k+2}(n) \ \leq \ \alpha_k(n) \ \leq \ \alpha'_k(n).$$

Otherwise, $n \geq k+4$. In this case, $\alpha'_{k+2}(n) = 2 + \alpha'_{k+2}(\alpha'_k(n))$ and $\alpha'_k(n) = 2 + \alpha'_k(\alpha'_{k-2}(n))$. Equation (A.1) and the first assertion of the induction hypothesis for the pair $(k, \alpha'_{k-2}(n))$ imply that $\alpha'_k(\alpha'_k(n)) \leq \alpha'_k(\alpha'_{k-2}(n))$. Also, Equation (A.1) and the third assertion of the induction hypothesis for the pair $(k, \alpha'_k(n))$ imply that $\alpha'_{k+2}(\alpha'_k(n)) \leq \alpha'_k(\alpha'_k(n))$. Thus, $\alpha'_{k+2}(\alpha'_k(n)) \leq \alpha'_k(\alpha'_k(n)) \leq \alpha'_k(\alpha'_{k-2}(n))$. Altogether,

$$\begin{aligned} \alpha'_{k+2}(n) &= 2 + \alpha'_{k+2}(\alpha'_k(n)) \\ &\leq 2 + \alpha'_k(\alpha'_{k-2}(n)) \ = \ \alpha'_k(n). \end{aligned}$$

B Proof of Lemma 2.4

This section is devoted to the proof of Lemma 2.4.

We start with proving the following lemma.

Lemma B.1. *For all $k \geq 0$ and $n \geq 0$, $\alpha_k(2(n+2)) < 2(\alpha_k(n)+2)$.*

Proof. The proof is by double induction on k and n. We restrict the attention to even values of k. The argument for odd values of k is similar, and is thus omitted.

Consider first the case $k = 0$. By definition, $\alpha_0(n) = \lceil n/2 \rceil$, for all $n \geq 0$. Hence, $\alpha_0(2(n+2)) = \lceil (2(n+2))/2 \rceil < 2(\lceil n/2 \rceil + 2) = 2(\alpha_0(n)+2)$.

Suppose next that $n < 2$. Note that $2(n+2) \leq 6$ and $\alpha_0(6) = \lceil 6/2 \rceil = 3$. By the third assertion of Lemma 2.1, $\alpha_k(6) \leq \alpha_0(6) = 3$. Hence, by the first assertion of Lemma 2.1, $\alpha_k(2(n+2)) \leq \alpha_k(6) \leq 3 < 2(\alpha_k(n)+2)$.

Induction Step: We assume that for an arbitrary pair (k, n), $n \geq 2$, $k \geq 2$, the statement holds for all pairs (k', n'), with either $k' < k$ or both $k' = k$ and $n' < n$, and prove it for the pair (k, n).

Since $n \geq 2$, we have $2(n+2) \geq 8$, and so Lemma 2.2 implies that $\alpha_k(2(n+2)) = 1 + \alpha_k(\alpha_{k-2}(2(n+2)))$. By the induction hypothesis for the pair $(k-2, n)$, we have $\alpha_{k-2}(2(n+2)) < 2(\alpha_{k-2}(n)+2)$. Hence, the first assertion of Lemma 2.1 implies that $\alpha_k(\alpha_{k-2}(2(n+2))) \leq \alpha_k(2(\alpha_{k-2}(n)+2))$. Since $n \geq 2$, the second assertion of Lemma 2.1 implies that $\alpha_{k-2}(n) < n$. Hence, by the induction hypothesis for the pair $(k, \alpha_{k-2}(n))$, it follows that $\alpha_k(2(\alpha_{k-2}(n)+2)) < 2(\alpha_k(\alpha_{k-2}(n))+2)$. Altogether,

$$\begin{aligned} \alpha_k(2(n+2)) &= 1 + \alpha_k(\alpha_{k-2}(2(n+2))) \\ &\leq 1 + \alpha_k(2(\alpha_{k-2}(n)+2)) \\ &< 1 + 2(\alpha_k(\alpha_{k-2}(n))+2) \\ &= 1 + 2(\alpha_k(n)+1) \\ &< 2(\alpha_k(n)+2). \qquad \blacksquare \end{aligned}$$

We are now ready to prove Lemma 2.4.

The first assertion of Lemma 2.3 implies that the function $\alpha'_k = \alpha'_k(n)$ is monotone non-decreasing with n, for all $k \geq 2$ and $n \geq 0$. The functions $\alpha'_0 = \lceil n/2 \rceil$ and $\alpha'_1 = \lceil \sqrt{n} \rceil$ are monotone non-decreasing with n as well. Consequently, $\alpha'_k(n) \leq \alpha'_k(2(n+2))$, for all $k \geq 0$ and $n \geq 0$. Hence, Lemma 2.4 follows as a corollary of the following lemma.

Lemma B.2. *For all $k \geq 0$ and $n \geq 0$, $\alpha'_k(2(n+2)) \leq 2(\alpha_k(n)+2)$.*

Proof. The proof is by double induction on k and n. We restrict the attention to even values of k. The argument for odd values of k is similar, and is thus omitted.

For the case $k = 0$, we have by definition $\alpha'_0(n) = \alpha_0(n)$, for all $n \geq 0$, and so the statement follows from Lemma B.1. We henceforth assume that $k \geq 2$.

Next, consider the case $n < 2$. In this case, $2(n+2) \leq 6$ and $\alpha_k(n) = 0$. Notice that $\alpha'_2(6) = 2 + \alpha'_2(3) = 2 + \alpha_2(3) = 4$. The third assertion of Lemma 2.3 implies that $\alpha'_k(6) \leq \alpha'_2(6)$. Hence, by the first assertion of Lemma 2.3, $\alpha'_k(2(n+2)) \leq \alpha'_k(6) \leq 4 = 2(\alpha_k(n)+2)$.

Suppose next that $2(n+2) \leq k+1$. In this case, we have by definition $\alpha'_k(2(n+2)) = \alpha_k(2(n+2))$, and so the statement follows from Lemma B.1.

Induction Step: We assume that for an arbitrary pair (k, n), $n \geq 2$, $k \geq 2$, $2(n+2) \geq k+2$, the statement holds for all pairs (k', n'), with either $k' < k$ or both $k' = k$ and $n' < n$, and prove it for the pair (k, n). Since $2(n+2) \geq k+2$, we have by definition $\alpha'_k(2(n+2)) = 2 + \alpha'_k(\alpha'_{k-2}(2(n+2)))$. By the induction hypothesis for the pair $(k-2, n)$, we have $\alpha'_{k-2}(2(n+2)) \leq 2(\alpha_{k-2}(n)+2)$. Hence, by the first assertion of Lemma 2.3, $\alpha'_k(\alpha'_{k-2}(2(n+2))) \leq \alpha'_k(2(\alpha_{k-2}(n)+2))$. Since $n \geq 2$, the second assertion of Lemma 2.1 yields $\alpha_{k-2}(n) < n$. Hence, the induction hypothesis for the pair $(k, \alpha_{k-2}(n))$ implies that $\alpha'_k(2(\alpha_{k-2}(n)+2)) \leq 2(\alpha_k(\alpha_{k-2}(n))+2)$. Altogether,

$$\begin{aligned} \alpha'_k(2(n+2)) &= 2 + \alpha'_k(\alpha'_{k-2}(2(n+2))) \\ &\leq 2 + \alpha'_k(2(\alpha_{k-2}(n)+2)) \\ &\leq 2 + 2(\alpha_k(\alpha_{k-2}(n))+2) \\ &= 2 + 2(\alpha_k(n)+1) \\ &= 2(\alpha_k(n)+2). \qquad \blacksquare \end{aligned}$$