

Low-Light Trees, and Tight Lower Bounds for Euclidean Spanners

Yefim Dinitz * Michael Elkin*[†] Shay Solomon*[†]

Abstract

We show that for every n -point metric space M and positive integer k , there exists a spanning tree T with unweighted diameter $O(k)$ and weight $w(T) = O(k \cdot n^{1/k}) \cdot w(MST(M))$, and a spanning tree T' with weight $w(T') = O(k) \cdot w(MST(M))$ and unweighted diameter $O(k \cdot n^{1/k})$. These trees also achieve an optimal maximum degree. Furthermore, we demonstrate that these trees can be constructed efficiently.

We prove that these tradeoffs between unweighted diameter and weight are *tight up to constant factors* in the entire range of parameters. Moreover, our lower bounds apply to a basic 1-dimensional Euclidean space.

Our lower bounds for the particular case of unweighted diameter $O(\log n)$ are of independent interest, settling a long-standing open problem in Computational Geometry. In STOC'95 Arya et al. devised a construction of Euclidean Spanners with unweighted diameter $O(\log n)$ and weight $O(\log n) \cdot w(MST(M))$. In SODA'05 Agarwal et al. showed that this result is tight up to a factor of $O(\log \log n)$. We close this gap and show that the result of Arya et al. is tight up to constant factors.

Finally, our upper bounds imply improved approximation algorithms for the *minimum h -hop spanning tree* and *bounded diameter minimum spanning tree* problems for metric spaces.

*Department of Computer Science, Ben-Gurion University of the Negev, POB 653, Beer-Sheva 84105, Israel. E-mail: {dinitz,elkinm,shayso}@cs.bgu.ac.il

Partially supported by the Lynn and William Frankel Center for Computer Sciences.

[†]This research has been supported by the Israeli Academy of Science, grant 483/06.

1 Introduction

1.1 Background and Main Results

Spanning trees for finite metric spaces have been a subject of an ongoing intensive research since the beginning of the nineties [3, 11, 12, 18, 31, 28, 13, 37, 10, 46, 9, 49]. In particular, many researchers studied the notion of *shallow-light trees*, henceforth SLTs [13, 37, 10, 9, 49, 4, 46]. Roughly speaking, an SLT of an n -point metric space M is a spanning tree T of the complete graph corresponding to M whose total weight is close to the weight $w(MST(M))$ of the minimum spanning tree $MST(M)$ of M , and whose weighted diameter is close to that of M . (See Section 2 for relevant definitions.)

In addition to being an appealing combinatorial object, SLTs turned out to be useful for various data gathering and dissemination problems in the message-passing model of distributed computing [10], in approximation algorithms [49], for constructing spanners [9, 4], and for VLSI-circuit design [23, 25, 24]. Near-optimal tradeoffs between the weight and diameter of SLTs were established by Awerbuch et al. [9], and by Khuller et al. [39].

Even though the requirement that the spanning tree T will have a small weighted-diameter is a natural one, it is no less natural to require it to have a small *unweighted diameter* (also called *hop-diameter*). The latter requirement guarantees that any two points of the metric space will be connected in T by a path that consists of only a small *number of edges* or *hops*. This guarantee turns out to be particularly important for routing [36, 1], computing almost shortest paths [21, 22], and in other applications. Another parameter that plays an important role in many applications is the maximum (vertex) degree of the constructed tree [6, 14, 8, 36].

In this paper we investigate a related notion of *low-light trees*, henceforth LLTs, that combine small weight with small hop-diameter. We present near-tight upper and lower bounds on the parameters of LLTs. In addition, our constructions of LLTs have *optimal maximum degree*.

To specify our results, we need some notation. For a spanning subgraph G of a metric space M , let $H = H(G)$ denote the hop-diameter of G , and $\Psi = \Psi(G) = \frac{w(G)}{w(MST(M))}$ denote the ratio between its weight and the weight of the minimum spanning tree of M , henceforth the *lightness* of G . The *hop-radius* $h(G, rt)$ of G with respect to a distinguished vertex rt is the maximum unweighted distance between rt and some vertex v in G . Obviously, $h(G, rt) \leq H(G) \leq 2 \cdot h(G, rt)$. For a rooted tree (T, rt) , the hop-radius (called also *depth*) of (T, rt) , denoted $h(T)$, is the hop-radius of T with respect to rt . The hop-radius of G , denoted $h(G)$, is defined by $h(G) = \min\{h(G, rt) \mid rt \in V\}$.

We show the following bounds that are tight up to constant factors in the *entire* range of the parameters.

1. For any sufficiently large integer n and positive integer h , and *any* n -point metric space M , there exists a spanning tree of M with hop-radius at most h and lightness at most $O(\Psi)$, for Ψ that satisfies the following relationship. If $h \geq \log n$ then $(h = O(\Psi \cdot n^{1/\Psi})$ and $\Psi = O(\log n)$). In the complementary range $h < \log n$, it holds that $\Psi = O(h \cdot n^{1/h})$.
Moreover, this spanning tree is a binary one whenever $h \geq \log n$, and it has the *optimal* maximum degree $\lceil n^{1/h} \rceil + 1$ whenever $h < \log n$. In addition, in the entire range of parameters the respective spanning trees can be constructed in optimal time $O(n^2)$.
2. For n and h as above, and $h \geq \log n$, there exists an n -point metric space $M^* = M^*(n)$ for which any spanning subgraph with hop-radius at most h has lightness at least $\Omega(\Psi)$, for some Ψ satisfying $h = \Omega(\Psi \cdot n^{1/\Psi})$.
3. For n and h as above, and $h < \log n$, any spanning subgraph with hop-radius at most h for $M^*(n)$ (see item 2) has lightness at least $\Psi = \Omega(h \cdot n^{1/h})$.

(Note that the equation $x \cdot n^{1/x} = \Theta(\log n)$ holds if and only if $x = \Theta(\log n)$.) See Figure 1 for an illustration of our results.

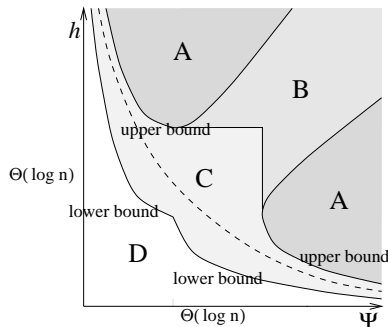


Figure 1: The dashed line separates two sets of pairs (Ψ, h) . For a pair (Ψ, h) above the line, for any n -point metric space there exists a spanning tree with lightness at most Ψ and hop-radius at most h . For a pair (Ψ, h) below the line, there exist n -point metric spaces for which this property does not hold. The two areas A and the area B are contained in the former set, while the area D is contained in the latter one. The two areas A depict our upper bound constructions, and their extension by monotonicity is depicted by the area B. The area D represents our lower bounds. The area C represents the gap between our upper and lower bounds.

The small maximum degree of our LLTs may be helpful for various applications in which the degree of a vertex v corresponds to the load on a processor that is located in v . The requirement to achieve small maximum degree is particularly important for applications in Computational Geometry. (See [6, 14, 8], and the references therein.)

Observe that to represent a general n -point metric space M one needs $O(n^2)$ space. Thus, the running time $O(n^2)$ of our algorithm is *linear* in the input size. Moreover, there is a variant of our algorithm that runs in time $O(n \cdot \log n)$ for *Euclidean* n -point metric spaces of any constant dimension.

We remark that our constructions of LLTs apply to metric spaces, and not to general graphs. We show that there are graphs with constant hop-diameter for which any spanning tree has either huge hop-diameter or huge weight, and thus our constructions cannot be extended to general graphs.

1.2 Lower Bounds for Euclidean Spanners

While our upper bounds apply to all finite metric spaces, our lower bounds apply to an extremely basic metric space $M^* = \vartheta_n$. Specifically, this metric space is the 1-dimensional Euclidean space with n points v_1, v_2, \dots, v_n lying on the x -axis with coordinates $1, 2, \dots, n$, respectively. The basic nature of ϑ_n strengthens our lower bounds, as they are applicable even for very limited classes of metric spaces. One particularly important application of our lower bounds is in the area of Euclidean Spanners. For a set \mathcal{U} of n points in \mathbb{R}^2 , and a parameter α , $\alpha \geq 1$, a subset \mathcal{H} of the $\binom{n}{2}$ segments connecting pairs of points from \mathcal{U} is called an (Euclidean) α -*spanner* for \mathcal{U} , if for every pair of points $u, v \in \mathcal{U}$, the distance between them in \mathcal{H} is at most α times the Euclidean distance between them in the plane. Euclidean spanner is a very fundamental geometric construct with numerous applications in Computational Geometry [6, 7, 8] and Network Design [36, 43]. (See the recent book of Narasimhan and Smid [45] for a detailed account on Euclidean spanners and their applications.)

A seminal paper that was a culmination of a long line of research on Euclidean spanners was published by Arya et al. [6] in STOC'95. One of the main results of this paper is a construction of $(1 + \epsilon)$ -spanners with $O(n)$ edges that also have lightness and hop-diameter both bounded by $O(\log n)$. As an evidence of the optimality of this combination of parameters, Arya et al. cited a result by Lenhof et al. [42]. Lenhof et

al. showed that any construction of Euclidean spanners that employs well-separated pair decompositions cannot achieve a better combination of weight and hop-diameter. However, the fundamental question of whether this combination of parameters can be improved by other means was left open in Arya et al. [6]. A partial answer to this intriguing problem was given by Agarwal et al. [2] in SODA'05. Specifically, it is shown in [2] that any Euclidean spanner of ϑ_n with lightness (respectively, hop-diameter) $O(\log n)$ must have hop-diameter (resp., lightness) at least $\Omega(\frac{\log n}{\log \log n})$. Consequently, Agarwal et al. showed that the upper bound of Arya et al. is optimal up to a factor of $O(\log \log n)$. The problem of closing this gap is stated explicitly as an open problem in the book of Narasimhan and Smid ([45], open problem 18 on p. 480). A simple corollary of our lower bounds is that the result of Arya et al. is tight up to constants even for *one-dimensional* spanners! In other words, we show that if the lightness (respectively, hop-diameter) is $O(\log n)$ then the hop-diameter (resp., lightness) is $\Omega(\log n)$, settling the open problem of [6, 2, 42, 45].

1.3 The h -Hop MST Problem

Our construction of LLTs can be also used to derive improved approximation algorithms for the *bounded diameter MST* (henceforth BDMST) and the *h -hop MST* problems for metrics. In these problems we are given a metric M and a positive integer h . The objective in the BDMST problem is to minimize the weight of a spanning tree T of M with hop-diameter at most h . In the closely related h -hop MST problem the objective is to minimize the weight of a rooted spanning tree with hop-radius at most h . Both problems are among the classical and most well-studied NP-hard problems. (See the book of Garey and Johnson [33], page 206, and [40, 17, 5, 38, 41, 34].) Kortsarz and Peleg [40] and Charikar et al. [17] devised an $O(\log n)$ -approximation algorithm for these problems when h is a constant, and an $O(n^\epsilon)$ -approximation algorithm for an arbitrarily small $\epsilon > 0$, that is applicable for all values of h . (These algorithms provide the same approximation guarantees for significantly more general problems.) Althaus et al. [5] devised a randomized $O(\log n)$ -approximation algorithm for all values of h , but the algorithm of [5] requires a very high running time of $O(n^5 \cdot h)$. Kantor and Peleg [38] devised $2^{O(h)}$ -approximation algorithms for these problems. Laue and Matijevic [41] presented a PTAS for 2-dimensional Euclidean metrics when h is a constant. These problems were also studied empirically. (See [34], and the references therein.)

Our constructions of LLTs give rise directly to improved approximation guarantees for these problems in the range $h = \Omega(\log n)$. Specifically, our approximation guarantee is $O(\Psi)$, where Ψ satisfies ($\Psi = O(\log n)$ and $h = O(\Psi \cdot n^{1/\Psi})$). In particular, for $h = \Theta(\log n)$ we obtain a deterministic $O(\log n)$ -approximation algorithm. Though the approximation guarantee of our algorithm for $h = \Theta(\log n)$ is the same as that of Althaus et al. [5], its running time $O(n^2)$ is drastically better than the running time $O(n^5 \cdot \log n)$ of the algorithm of [5]. In addition, the algorithm of [5] is randomized, while our algorithm is deterministic. Moreover, for $\omega(\log n) = h = O(n)$ the approximation guarantee of our algorithm becomes *sublogarithmic*. Thus, in this range our algorithm improves the current state-of-the-art both in terms of the approximation guarantee and running time. Finally, when $h = n^\epsilon$, for some constant $\epsilon > 0$, our approximation guarantee becomes *constant*. See also Table 1.

To summarize, the problem of understanding the inherent tradeoff between different parameters of LLTs is a fundamental one in the investigation of spanning trees for metric spaces. In addition, this basic and combinatorially appealing problem has important applications in Computational Geometry and Approximation Algorithms. We believe that further investigation of LLTs will expose their additional applications, and connections to other areas.

	$h = \Theta(\log n)$	$h = \log^c n, c > 1$	$h = 2^{c\sqrt{\log n}}, c > 0$	$h = n^\epsilon, \epsilon > 0$
Previous [5] approximation	$O(\log n)$ randomized	$O(\log n)$ randomized	$O(\log n)$ randomized	$O(\log n)$ randomized
Our approximation	$O(\log n)$ deterministic	$O((\log n)/(\log \log n))$ deterministic	$O(\sqrt{\log n})$ deterministic	$O(\epsilon^{-1})$ deterministic
Previous [5] runtime	$O(n^5 \cdot h)$	$O(n^5 \cdot h)$	$O(n^5 \cdot h)$	$O(n^5 \cdot h)$
Our runtime	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$

Table 1: A summary of previous and our results for the h -hop MST and BDMST problems for metric spaces, for different values of h . Our results are indicated by bold font.

1.4 Overview and Our Techniques

The most technically challenging part of our proof is the lower bound for the range of $h \geq \log n$. The proof of this lower bound consists of a number of components. First, we restrict our attention to binary trees. Second, we adapt a linear program for the minimum linear arrangement problem from the seminal paper of Even, Naor, Rao and Schieber [30] on spreading metrics to our needs. Third, we analyze this linear program and show that the problem of providing a lower bound for its solution reduces to a clean combinatorial problem, and solve this problem. This enables us to establish the desired lower bounds for *binary trees*. Finally, we extend these lower bounds to general trees. This part of our proof demonstrates that techniques from the area of Low Distortion Embeddings (such as linear programs based on spreading metrics) may be useful for proving lower bounds in Computational Geometry. We anticipate that our approach will be applicable to other open problems in this area.

The proof of our lower bounds for $h < \log n$ combines some ideas from Agarwal et al. [2] with numerous new ideas. Specifically, Agarwal et al. reduce the problem from the general family of spanning subgraphs for ϑ_n to a certain restricted family of *stack graphs*. This reduction of [2] provides a very elegant way for achieving somewhat weaker bounds, but it is inherently suboptimal. In our proof we tackle the general family of graphs directly. This more direct approach results in a much more technically involved proof on the one hand, and in much more accurate bounds on the other.

For upper bounds we essentially reduce the problem of constructing LLTs for general metric spaces to the same problem on ϑ_n . Somewhat surprisingly, despite the apparent simplicity of the metric space ϑ_n , the problem of constructing LLTs for this space appears to be quite non-trivial.

1.5 Related Work

SLTs have been extensively studied for the last twenty years [13, 37, 10, 9, 23, 25, 24, 39, 4]. However, all these constructions of SLTs may result in trees with very large hop-diameter, and the techniques used in these constructions appear to be inapplicable to the problem of constructing LLTs.

Euclidean spanners are also a subject of a recent extensive and intensive research (see [6, 27, 2, 7], and the references therein). However, the basic technique for constructing them relies heavily on the methodology of well-separated pair decomposition due to Callahan and Kosaraju [15]. This powerful methodology is, however, applicable only for Euclidean metric spaces of constant dimension, while our constructions apply to general metric spaces.

Tight lower bounds on the hop-diameter of Euclidean spanners with a given number of edges have been recently established by Chan and Gupta [16]. Specifically, it is shown in [16] that for any $\epsilon > 0$ there exists an n -point Euclidean metric space $M = M(n, \epsilon)$ for which any Euclidean $(1 + \epsilon)$ -spanner with

m edges has hop-diameter $\Omega(\alpha(m, n))$, where α is the functional inverse of the Ackermann's function. Moreover, the metric space M is 1-dimensional. (On the other hand, the space M is still not as restricted as ϑ_n .) However, this lower bound provides no indication whatsoever as to how *light* can be Euclidean spanners with low hop-diameter. In particular, the construction of Arya et al. [6] that provides matching upper bounds to the lower bounds of [16] produces spanners that may have very large weight.

In terms of the techniques, Chan and Gupta [16] start with proving their lower bounds for metrics induced by binary hierarchically-separated-trees (henceforth, HSTs), and then translate them into lower bounds for metrics induced by n points on the real line using known results. Their proof of the lower bound for HSTs is an extension of Yao's proof technique from [50]. As was discussed above, our lower bounds are achieved by completely different proof techniques that involve analyzing a linear program based on spreading metrics. In particular, our lower bounds are proved directly for ϑ_n .

The study of spanning trees of the 1-dimensional metric space ϑ_n is related to the extremely well-studied problem of computing partial-sums. (See the papers of Yao [50], Chazelle and Rosenberg [20], Pătraşcu and Demaine [47], and the references therein.) For a discussion about the relationship between these two problems we refer to the introduction of [2]. We anticipate that our techniques will be useful in proving lower bounds for the problem of computing partial sums as well.

The spreading metric linear program for the minimum linear arrangement problem that we use for our lower bounds was studied in [30, 48]. There is an extensive literature on the minimum linear arrangement problem itself. (See [19, 32], and the references therein).

There is a large body of literature dealing with bicriteria approximation algorithms (see, e.g., [44]). Despite the seeming similarity between the bicriteria setting and our setting, the bicriteria results are inapplicable to our problem. For an illustration, consider an algorithm from [44] that, given a metric space M and a positive integer h , constructs a spanning tree T of hop-diameter $O(\log n) \cdot h$ and weight $O(\log n) \cdot w(T_h^*)$, where T_h^* is the spanning tree of minimum weight among all spanning trees of hop-diameter at most h . If h is set to a constant, one obtains hop-diameter $O(\log n)$, but the weight bound of $O(\log n) \cdot w(T_h^*)$ may be much larger than $O(\log n) \cdot w(MST(M))$. On the other hand, if $h = h(n)$ tends to infinity as n grows, then the bound on hop-diameter will be greater than logarithmic. To summarize, the result of [44] does not imply the existence of an LLT with hop-diameter and lightness $O(\log n)$, which follows as an immediate corollary of our upper bounds.

1.6 The Structure of the Paper

In Section 2 we define the basic notions and present the notation that is used throughout the paper. Sections 3-5 are devoted to lower bounds. In Section 3 we analyze certain auxiliary functions defined in Section 2, and show that these functions are monotone non-increasing with the depth parameter. Their monotonicity is employed in Sections 4 and 5 for proving lower bounds. In Section 4.1 we analyze trees that have depth $h \geq \log n$. In Section 4.2 we turn to the complementary range $h < \log n$. In Section 5 we use the lower bounds for LLTs proven in Section 4 to derive our lower bounds on the tradeoff between the hop-diameter and weight for Euclidean spanners. Our upper bounds for LLTs are presented in Section 6. Some basic properties of the binomial coefficients that we use in our analysis appear in Appendix A. A simple argument precluding the existence of good LLTs for general *graphs* (in contrast to *metrics*) appears in Appendix B.

2 Preliminaries

For a positive integer n , an n -point metric space $M = (V, dist_M)$ can be viewed as the complete graph $G = G(M) = (V, \binom{V}{2}, dist_M)$ in which for every pair of vertices $u, w \in V$, the weight of the edge $e = (u, w)$ between u and w in G is defined by $w(u, w) = dist_M(u, w)$. The distance function $dist_M$ is required to be

non-negative, equal to zero when $u = w$, and to satisfy the triangle inequality ($dist_M(u, w) \leq dist_M(u, v) + dist_M(v, w)$, for every triple $u, w, v \in V$). A graph G' is called a *spanning subgraph* (respectively, *spanning tree*; *minimum spanning tree*) of M if it is a spanning subgraph (resp., spanning tree; minimum spanning tree) of $G(M)$.

For a weighted graph $G = (V, E, w)$, and a path P in G , its (*weighted*) *length* is defined as the sum of the weights of edges along P , and its *unweighted length* (or *hop-length*) is the number $|P|$ of edges (or *hops*) in P . For a pair of vertices $u, w \in V$, the *weighted distance* in G between u and w , denoted $dist_G(u, w)$, is the smallest weighted length of a path connecting between u and w in G . The *weighted* (respectively, *unweighted* or *hop-*) *diameter* of G is the maximum weighted (resp., unweighted) distance between a pair of vertices in V . For a rooted tree (T, rt) and a vertex v in T , the *level* of v in T is the unweighted distance between the root rt of T and v in T .

Whenever n can be understood from the context, we write ϑ as a shortcut for ϑ_n . We will use the notion ϑ -tree as an abbreviation for a “rooted spanning tree of ϑ ”. We say that an edge (v_i, v_j) connecting a parent vertex v_i with a child vertex v_j in a ϑ -tree is a *right* (respectively, *left*) *edge* if $i > j$ (resp., $i < j$). In this case v_j is called a *right* (resp., *left*) *child* of v_i . An edge (v_i, v_j) is said to *cover* a vertex v_ℓ if $i < \ell < j$. For a ϑ -tree T , the number of edges $e \in E(T)$ that cover a vertex v of ϑ is called the *covering of v by T* and it is denoted $\gamma(v) = \gamma_T(v)$. The *covering of the tree T* , $\gamma(T)$, is the maximum covering of a vertex v in ϑ by T , i.e.,

$$\gamma(T) = \max\{\gamma_T(v) \mid v \in V(\vartheta)\}.$$

For a pair of positive integers n and h , $1 \leq h \leq n - 1$, denote by $\gamma(n, h)$ (respectively, $W(n, h)$) the minimum covering (resp., weight) taken over all ϑ_n -trees of depth h . (See Section 1.1 for the definition of depth.) As was shown in [2], the notions of covering and lightness are closely related.

We use the notion of *abstract tree* to refer to a connected acyclic graph T . In contrast to a ϑ -tree, in an abstract tree the vertices are not necessarily labeled. A tree in which every vertex has at most d children, for a positive integer d , is called a *d -ary tree*.

For a pair of non-negative integers k, n , $k \leq n$, we denote the sets $\{k, k + 1, \dots, n\}$ and $\{1, 2, \dots, n\}$ by $[k, n]$ and $[n]$, respectively.

Finally, $\log x$ (respectively, $\ln x$) stands for the logarithm of x in base 2 (resp., e), $\log_2 x$ (resp., $\log_e x$).

3 Monotonicity of Weight and Covering

In this section we restrict our attention to ϑ -trees and show that both the covering and the weight does not increase as the tree depth grows. This property is very useful for proving lower bounds (cf. Section 4.2).

Fix a positive integer n . In what follows we write $\gamma(h)$ (respectively, $W(h)$) as a shortcut for $\gamma(n, h)$ (resp., $W(n, h)$).

Lemma 3.1 *The sequences $(\gamma(1), \gamma(2), \dots, \gamma(n - 1))$ and $(W(1), W(2), \dots, W(n - 1))$ are monotone non-increasing.*

Proof: We prove the statement for the sequence $(\gamma(1), \gamma(2), \dots, \gamma(n - 1))$. The proof that the sequence $(W(1), W(2), \dots, W(n - 1))$ is monotone as well is analogous. Note that $\gamma(n - 1)$ is equal to the covering $\gamma(P)$ of the n -vertex path $P = P_n = (v_1, v_2, \dots, v_n)$ rooted at v_1 . Since for every $i \in [n]$, $\gamma_P(v_i) = 0$, it follows that $\gamma(n - 1) = \gamma(P) = 0$. Observe also that for $h \geq 1$, $\gamma(h)$ is non-negative. Consequently, we henceforth restrict our attention to the subsequence $(\gamma(1), \gamma(2), \dots, \gamma(n - 2))$.

Let T be a ϑ -tree that has depth h , $1 \leq h \leq n - 2$ and covering $\gamma = \gamma(h)$. (In other words, the tree T has the minimum covering among all trees of depth equal to h .) We denote its root by rt . We construct

a tree $S(T)$ that has depth $h + 1$ and covering at most γ . Consider a vertex v at distance h from rt , and the path $P = (u_0 = rt, u_1, \dots, u_{h-1}, v)$ between them in T .

1. Since $h \leq n - 2$, there exists at least one leaf ℓ in T which is not in P . Remove ℓ along with the edge connecting it to its parent in T .
2. Let ϵ , $0 < \epsilon < 1$, be a small real value. Assume that $v > u_{h-1}$. Insert a new vertex v' , $v' = v + \epsilon$, to be a right child of v . (If $v < u_{h-1}$ then $v - \epsilon$ is inserted as a left child of v .)

Denote the resulting tree by T' . Note that $|V(T')| = |V(T) \setminus \{\ell\} \cup \{v'\}| = n$. Clearly, the first step neither changes the depth h nor increases the covering γ . Since the distance from rt to the farthest vertex v in T is h , adding v' as a new child of v in the second step increases the depth of the tree by exactly one. Note that since $\epsilon < 1$, the new edge (v, v') does not cover any vertex in T' , implying that the covering of any vertex $v \in V(T') \setminus \{v'\}$ in T' is no greater than its covering in T . To conclude that the covering of T' is at most γ , we show that the covering of the new vertex v' in T' is no greater than γ . In fact, we argue that any edge that covers v' also covers v in T' , which provides the required result. To see this, note that for an edge e that covers v' not to cover v , it must hold that e is incident to v . However, since v is a leaf in T , the only edges which are incident to v in T' are (u_{h-1}, v) and the new edge (v, v') , both of which do not cover v' , and we are done. (See Figure 2 for an illustration.)

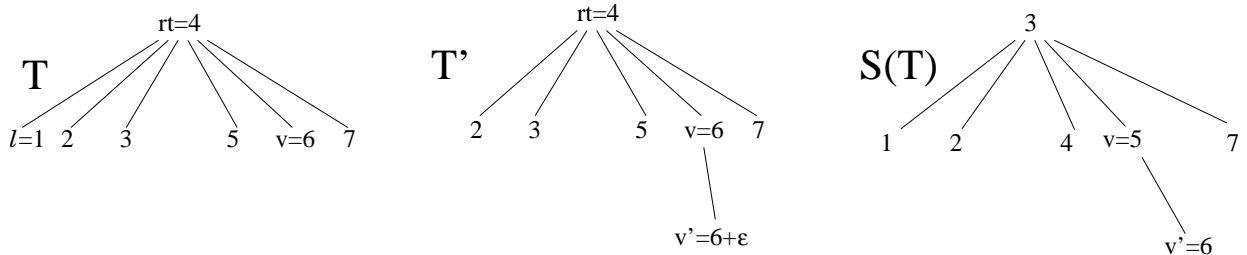


Figure 2: The ϑ_7 -tree T rooted at $rt = 4$ is depicted on the left. This tree has depth 1 and covering 2. The tree T' on the vertices $\{2, 3, 4, 5, 6, 6 + \epsilon, 7\}$ rooted at $rt = 4$ is depicted in the middle. It has depth 2 and covering 2. This tree is obtained from T by removing the vertex $\ell = 1$ along with the edge (rt, ℓ) , and adding the vertex $v' = (6 + \epsilon)$ along with the edge (v, v') . The ϑ_7 -tree $S(T)$ rooted at 3 is depicted on the right. It has depth 2 and covering 2. This tree is obtained from T' by relocating the points $2, 3, 4, 5, 6, 6 + \epsilon, 7$ to the points $1, 2, \dots, 7$, respectively.

Observe that T' does not span ϑ . Let $z_1 < z_2 < \dots < z_n$ be the sequence of vertices of T' in an increasing order. To transform T' into a spanning tree $S(T)$ of ϑ , for each index i , $1 \leq i \leq n$, relocate z_i to the point i .

Let T^* be a spanning tree that has depth $h + 1$ and minimum covering $\gamma(h + 1)$. By definition, $\gamma(h + 1)$ is no greater than the covering of $S(T)$, which is at most $\gamma(h)$. Hence $\gamma(h + 1) \leq \gamma(h)$, and we are done. ■

We remark that these monotonicity properties (Lemma 3.1) apply to any 1-dimensional Euclidean space (rather than just to ϑ).

4 Lower Bounds

In this section we devise lower bounds for lightness of ϑ -trees for the entire range of parameters. In Section 4.1 we analyze trees of depth $h \geq \log n$ (henceforth *high* trees), and in Section 4.2 we study trees

with depth in the complementary range $h < \log n$ (henceforth *low trees*).

4.1 Lower Bounds for High Trees

In this section we devise lower bounds for high ϑ -trees. In Sections 4.1.1 and 4.1.2 we restrict our attention to binary trees, reduce this restricted variant of the problem to a certain question concerning the minimum linear arrangement problem, and resolve the latter question. In Section 4.1.3 we show that the lower bound for binary trees extends to general trees, and, in fact, to general spanning subgraphs.

We start with introducing some terminology that will be used in this section. A binary abstract rooted tree on n vertices of depth at most h will be called an (n, h) -tree. Let $B_n(h)$ denote the family of all (n, h) -trees, and denote by $Bin(n, h)$ the minimum weight of a ϑ -tree in $B_n(h)$. Consider a rooted tree $T = (T, rt)$ in $B_n(h)$. For a vertex v in T , let T_v be the subtree of T rooted at v , and U_v be its vertex set. The depth of a vertex v in T is the depth of the subtree T_v of T rooted at v .

4.1.1 The Minimum Linear Arrangement Problem

In this section we describe a relationship between the problem of constructing LLTs and the *minimum linear arrangement* (henceforth, *MINLA*) problem [30, 48]. It turns out that lower bounds for the value of the objective function of the MINLA problem for abstract trees give rise to analogous lower bounds for the tradeoff between the depth and lightness of LLTs.

The MINLA problem is defined as follows. Given an undirected graph $G = (V, E)$, we would like to find a permutation (called also a *linear arrangement*) of the nodes $\sigma : V \rightarrow \{1, \dots, n = |V|\}$ that minimizes the cost of the linear arrangement σ ,

$$LA(G, \sigma) = \sum_{(i,j) \in E} |\sigma(i) - \sigma(j)|.$$

The minimum linear arrangement of the graph G , denoted $MINLA(G)$, is defined as the minimum cost of a linear arrangement, i.e.,

$$MINLA(G) = \min\{LA(G, \sigma) \mid \sigma \in S_n\},$$

where S_n is the set of all permutations of $[n]$.

Let $G = (V, E)$ be an n -vertex graph. For a permutation $\sigma \in S_n$, let $G_\sigma = ([n], E_\sigma)$ denote the graph with vertex set $[n]$ and edge set $E_\sigma = \{(\sigma(u), \sigma(w)) \mid (u, w) \in E\}$, equipped with the weight function $w(i, j) = |i - j|$ for every $i \neq j, i, j \in [n]$. (G_σ is an isomorphic copy of G .) Observe that $LA(G, \sigma) = w(G_\sigma) = \sum_{e \in E_\sigma} w(e)$. Also, let $G^* = ([n], E^*)$ denote the graph G_{σ^*} for the *optimal* permutation $\sigma^* = \sigma(G)$, that is, for σ^* such that $w(G_{\sigma^*}) = \min\{w(G_\sigma) \mid \sigma \in S_n\}$. It follows that $MINLA(G)$ is equal to $w(G_{\sigma^*})$. Moreover, for a family \mathcal{F} of n -vertex graphs, the minimum weight $\{w(G_{\sigma^*}) \mid G \in \mathcal{F}\}$ is precisely equal to the minimum value $\{MINLA(G) \mid G \in \mathcal{F}\}$ of the MINLA problem on one of the graphs of the family \mathcal{F} . Next, we show a *lower bound on the value*

$$MINLA(B_n(h)) = \min\{MINLA(G) \mid G \in B_n(h)\}$$

of the MINLA problem for graphs of the family $B_n(h)$. By the above considerations this lower bound will apply also to the minimum weight $Bin(n, h)$ of a ϑ -tree from $B_n(h)$.

This lower bound is established by analyzing the following linear program relaxation *LP1* for the MINLA problem. This relaxation was introduced and studied in a seminal work on spreading metrics by Even et al. [30] The variables of this linear program $\{\ell(e) \mid e \in E\}$ can be viewed as edge lengths. For a pair of vertices u and v , $dist_\ell(u, v)$ stands for the distance between u and v in the graph G equipped

with the length function $\ell(\cdot)$ on its edges.

$$\begin{aligned}
LP1 : \quad & \min \sum_{e \in E} \ell(e) \\
s.t. \quad & \forall U \subseteq V, \forall v \in U : \sum_{u \in U} \text{dist}_\ell(v, u) \geq \frac{1}{4} \cdot (|U|^2 - 1) \\
& \forall e \in E : \ell(e) \geq 0.
\end{aligned}$$

It is well-known that the *optimal value of LP1, denoted $OPT(LP1)$, is a lower bound on $MINLA(G)$ [30, 48].*

Next, we present a variant $LP2$ of $LP1$ which involves only a small subset of constraints that are used in $LP1$. Clearly, *the optimal value of $LP2$, denoted $OPT(LP2)$, is a lower bound on that of $LP1$.* While in $LP1$ there is a constraint for each pair (U, v) , $U \subseteq V$, $v \in U$, there are only the constraints that correspond to pairs (U_v, v) present in $LP2$ for T .

$$\begin{aligned}
LP2 : \quad & \min \sum_{e \in E(T)} \ell(e) \\
s.t. \quad & \forall v \in V : \sum_{u \in U_v} \text{dist}_\ell(u, v) \geq \frac{1}{4} \cdot (|U_v|^2 - 1) \\
& \forall e \in E : \ell(e) \geq 0.
\end{aligned}$$

We will henceforth use the shortcut $\text{dist}(u, v)$ for $\text{dist}_\ell(u, v)$.

Let $\text{Ineq}(v)$ be the inequality $\sum_{u \in U_v} \text{dist}(v, u) \geq \frac{1}{4}(|U_v|^2 - 1)$, and $\text{Eq}(v)$ be the corresponding equation $\sum_{u \in U_v} \text{dist}(v, u) = \frac{1}{4}(|U_v|^2 - 1)$. We use the notation $\text{TotalDist}(v)$ to denote the left-hand side of $\text{Ineq}(v)$, i.e., $\text{TotalDist}(v) = \sum_{u \in U_v} \text{dist}(v, u)$.

Note that the trees we consider in the context of the MINLA problem are *abstract* rooted n -vertex trees, and not ϑ -trees. In a binary abstract rooted tree T , we use the terms *junior* and *senior* instead of “left” and “right”, respectively, when referring to the children of an inner vertex v . (We avoid using the more standard terminology of “left” and “right” children to prevent ambiguity with ϑ -trees.) The junior (respectively, senior) child of an inner vertex v is denoted by $v.jr$ (resp., $v.sr$).

We replace all inequalities $\text{Ineq}(v)$ by equations $\text{Eq}(v)$ in $LP2$. By the following lemma, *the optimal value of $LP2$ remains intact.*

Lemma 4.1 *For a binary tree T , in any optimal solution to $LP2$ all inequalities $\{\text{Ineq}(v) \mid v \in V\}$ hold as equalities.*

Proof: First, observe that for a leaf z in T ,

$$\text{TotalDist}(z) = \sum_{u \in U_z} \text{dist}(u, z) = 0,$$

implying that $\text{Ineq}(z)$ holds as equality.

Let n denote the number of vertices of T . Order the $(n-1)$ edges e_1, e_2, \dots, e_{n-1} arbitrarily, and consider the subset \mathcal{C} of all value assignments ψ to the variables $\ell(e_1), \ell(e_2), \dots, \ell(e_{n-1})$, that constitute an optimal solution to the linear program $LP2$, and such that there exists a vertex $v \in V(T)$ for which $\text{Ineq}(v)$ holds as a strict inequality under ψ . Suppose for contradiction that $\mathcal{C} \neq \emptyset$. For an assignment $\psi \in \mathcal{C}$, let the

level of ψ , denoted $L(\psi)$, be the minimum level of a vertex v in T for which $Ineq(v)$ holds as a strict inequality. Consider an optimal solution $\psi^* \in \mathcal{C}$ of minimum level, that is,

$$L(\psi^*) = \min\{L(\psi) \mid \psi \in \mathcal{C}\}.$$

Let v be an inner vertex of level $L(\psi^*)$ for which $Ineq(v)$ holds as a strict inequality under the assignment ψ^* . By definition,

$$TotalDist(v) > \frac{1}{4}(|U_v|^2 - 1).$$

It is convenient to imagine that the vertices of T_v are colored in two colors as follows. The root vertex v of T_v is colored white. All leaves are colored black. An inner vertex $u \in U_v \setminus \{v\}$ is colored white, if the following three conditions hold.

- Its parent $\pi(u)$ in T_v is colored white.
- $Ineq(u)$ holds as a strict inequality.
- All edges e connecting u to its children satisfy $\ell(e) = 0$ under ψ^* .

Otherwise, u is colored black. (See Figure 3 for an illustration.)

Remark: Observe that for a white vertex $u \in U_v \setminus \{v\}$, all vertices of the path $P_{v,\pi(u)}$ connecting v with $\pi(u)$ are colored white.

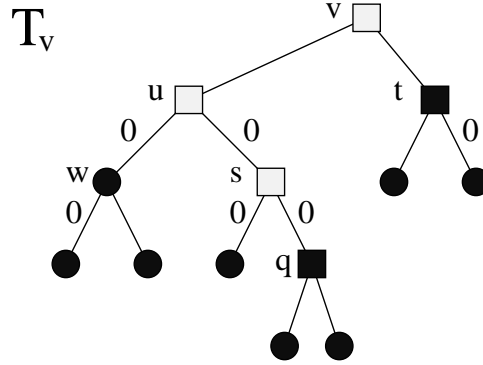


Figure 3: This figure depicts a white-black coloring of the subtree T_v . Vertices satisfying the second condition are depicted as squares, while all other vertices are depicted as circles. Positive edge weights do not appear in the figure. The vertices u and s are colored white as they satisfy all three conditions. The inner vertex w is colored black as it is not a square. The inner vertices t and q are colored black as they do not satisfy the third condition. The white vertex of minimum depth in T_v is s ; its depth is 2.

Claim 4.2 *At least one vertex of depth 1 in T_v is colored white.*

Proof: Suppose for contradiction that all white vertices in T_v have depth at least 2. Let w be a white vertex of minimum depth d , $d \geq 2$. Since w is colored white, all vertices in the path $P_{v,w}$ connecting v with w in T_v are colored white as well, implying that for each vertex x along that path, $Ineq(x)$ holds as a strict inequality, and all edges that connect x to its children have weight zero.

First, we assume that w has only one child. Suppose without loss of generality that it is $w.jr$. Since the edge $(w, w.jr)$ have weight zero, it holds that $TotalDist(w) = TotalDist(w.jr)$. Since $Ineq(w)$ holds as a strict inequality,

$$TotalDist(w) > \frac{1}{4} \cdot (|U_w|^2 - 1) = \frac{1}{4} \cdot ((|U_{w.jr}| + 1)^2 - 1) > \frac{1}{4} \cdot (|U_{w.jr}|^2 - 1),$$

implying that $Ineq(w.jr)$ holds as a strict inequality.

Next, we assume that w has two children. Since the edges that connect w to its children have weight zero, it holds that

$$TotalDist(w) = TotalDist(w.jr) + TotalDist(w.sr).$$

Since $Ineq(w)$ holds as a strict inequality,

$$\begin{aligned} TotalDist(w) &> \frac{1}{4} \cdot (|U_w|^2 - 1) = \frac{1}{4} \cdot (|U_{w.jr}| + |U_{w.sr}| + 1)^2 - 1 \\ &> \frac{1}{4} \cdot (|U_{w.jr}|^2 - 1) + \frac{1}{4} \cdot (|U_{w.sr}|^2 - 1). \end{aligned}$$

Thus, at least one among the two inequalities $Ineq(w.jr)$ and $Ineq(w.sr)$ holds as a strict one. We assume without loss of generality that $Ineq(w.jr)$ holds as a strict inequality.

Since for any leaf z , $Ineq(z)$ holds as equality, it follows that $w.jr$ is not a leaf.

To complete the proof of Claim 4.2 we need the following claim.

Claim 4.3 *All edges that connect $w.jr$ to its children have value zero under ψ^* .*

Proof: Suppose for contradiction that there is a child y of $w.jr$ such that the length of the edge $e = (w.jr, y)$ under the assignment ψ^* is some $\delta > 0$. Consider the path $P_{v,w} = (v_0 = v, v_1, \dots, v_j = w)$, $j \geq 0$, connecting the vertices v and w . The analysis splits into two cases depending on whether v is the root rt of T or not. First, suppose that $v = rt$. Observe that for every index i , $i \in [0, j]$,

$$f_i(\delta) = f_i(\ell(e)) = TotalDist(v_i) - \frac{1}{4} \cdot (|U_{v_i}|^2 - 1)$$

is a continuous function of the variable $\ell(e)$. Since for every $i \in [0, j]$, $f_i(\delta) > 0$, we can slightly decrease the value of $\ell(e)$ and set it to some δ' , $0 < \delta' < \delta$, so that all $f_i(\delta')$ are still non-negative. However, this change in the value of $\ell(e)$ results in a new feasible assignment ψ' of values to the variables $\{\ell(e) \mid e \in E(T)\}$. Moreover, obviously $\sum_{e \in E(T)} \ell(e)$ of the objective function of $LP2$ is smaller under ψ' than under ψ^* . This is a contradiction to the assumption that ψ^* is an optimal solution for $LP2$.

The case that $v \neq rt$ is handled similarly. In this case the difference $\epsilon = \delta - \delta'$ (with δ' defined exactly as in the first case) is added to the value of $\ell(e')$, for the edge $e' = (v, \pi(v))$ connecting v to its parent in T . (See Figure 4 for an illustration.) It is easy to verify that the resulting assignment $\tilde{\psi}$ is feasible, and that the value of the objective function $\sum_{e \in E(T)} \ell(e)$ is the same under ψ^* and $\tilde{\psi}$. Also, since for every $i \in [0, j]$, $f_i(\ell(e)) = f_i(\delta') > 0$ under $\tilde{\psi}$, it follows that the inequalities $Ineq(v_i)$ hold as strict inequalities for all $i \in [0, j]$, and thus both assignments $\tilde{\psi}$ and ψ^* belong to the set \mathcal{C} . However, $Ineq(\pi(v))$ holds as a strict inequality under $\tilde{\psi}$ as well, and thus $L(\tilde{\psi}) < L(\psi^*)$. This is a contradiction to the assumption that ψ^* has the minimum level in \mathcal{C} . Hence under ψ^* , all edges that connect $w.jr$ to its children have value zero. ■

Recall that $Ineq(w.jr)$ holds as a strict inequality, and $w = \pi(w.jr)$ is a white vertex. Consequently, $w.jr$ should be colored white as well. However, its depth is smaller than the minimum depth of a white vertex in T_v , contradiction. This completes the proof of Claim 4.2. ■

Consider a white vertex x in T_v of depth 1. The edges connecting x to its children are assigned value zero, implying that $TotalDist(x) = 0$. However, since x is colored white, the inequality $Ineq(x)$ holds as a strict inequality, i.e.,

$$TotalDist(x) > \frac{1}{4} \cdot (|U_x|^2 - 1) > 0.$$

This is a contradiction to the assumption that \mathcal{C} is not empty, proving Lemma 4.1. ■

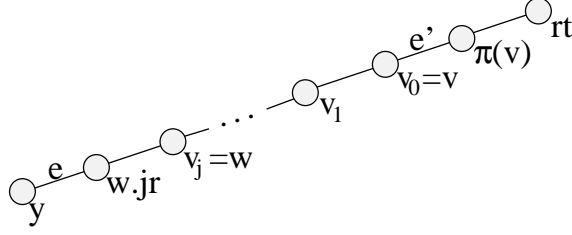


Figure 4: The path $(rt, \pi(v), v_0 = v, v_1, \dots, v_j = w, w.jr, y)$. The weight of e is slightly decreased, and this is compensated by increasing the weight of e' .

Observe that in an abstract rooted tree T , children's seniority has no effect on the value of $MINLA(T)$. Hence, any binary abstract rooted tree T can be restructured to guarantee that for every inner vertex v in it, the subtree rooted at $v.jr$ is at least as large (in terms of the number of vertices) as the subtree rooted at $v.sr$. We refer to this restructuring process as the *senior-adjustment* of T . We say that a binary abstract rooted tree \tilde{T} that satisfies the condition above for every inner vertex is *senior-adjusted*. (See Figure 5 for an illustration.) In the remainder of this section, we restrict our attention (without loss of generality) to senior-adjusted trees.

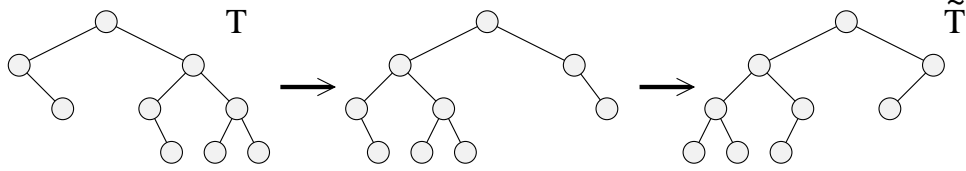


Figure 5: The original rooted tree T is restructured to produce the senior-adjusted tree \tilde{T} . For illustrative purposes, a junior (respectively, senior) child is drawn to the left (resp., right) of its parent.

Consider a subtree T_v rooted at an inner vertex v . The vertex v has a junior child $v.jr$, and possibly a senior child $v.sr$, each being the root of the corresponding subtrees $T_{v.jr}$ and $T_{v.sr}$, respectively. Let $U_{jr} = U_{v.jr}$, $U_{sr} = U_{v.sr}$, $n_{jr} = |U_{jr}|$, $n_{sr} = |U_{sr}|$, $e_{jr} = (v, v.jr)$, $e_{sr} = (v, v.sr)$, $x_{jr} = \ell(e_{jr})$, and $x_{sr} = \ell(e_{sr})$. If v has only a junior child, then we write $v.sr = T_{v.sr} = U_{v.sr} = NULL$, and $n_{sr} = x_{sr} = 0$. Since we restrict our attention to senior-adjusted trees, $n_{jr} \geq n_{sr}$ for any v .

The next lemma provides a lower bound on the sum $x_{jr} + x_{sr}$ of values assigned by an optimal solution for $LP2$ to the edges e_{jr} and e_{sr} .

Lemma 4.4 *For an optimal solution for $LP2$,*

$$x_{jr} + x_{sr} > \frac{1}{2} \cdot (\min\{n_{jr}, n_{sr}\} + 1) = \frac{1}{2} \cdot (n_{sr} + 1).$$

Proof: It is easy to verify that

$$\sum_{u \in U_v} \text{dist}(v, u) = x_{jr} \cdot n_{jr} + \sum_{u \in U_{jr}} \text{dist}(v.jr, u) + x_{sr} \cdot n_{sr} + \sum_{u \in U_{sr}} \text{dist}(v.sr, u). \quad (1)$$

By Lemma 4.1, both inequalities $Ineq(v)$ and $Ineq(v.jr)$ hold as equalities, i.e.,

$$\sum_{u \in U_v} \text{dist}(v, u) = \frac{1}{4} \cdot (|U_v|^2 - 1) = \frac{1}{4} \cdot ((n_{jr} + n_{sr} + 1)^2 - 1). \quad (2)$$

$$\sum_{u \in U_{jr}} \text{dist}(v.jr, u) = \frac{1}{4} \cdot (n_{jr}^2 - 1). \quad (3)$$

The analysis splits into two cases.

Case 1: v has two children. By Lemma 4.1, the inequality $\text{Ineq}(v.sr)$ holds as equality as well, i.e.,

$$\sum_{u \in U_{sr}} \text{dist}(v.sr, u) = \frac{1}{4} \cdot (n_{sr}^2 - 1). \quad (4)$$

Plugging (2), (3), and (4) in (1) implies

$$x_{jr} \cdot n_{jr} + x_{sr} \cdot n_{sr} = \frac{1}{2} \cdot (n_{jr} \cdot n_{sr} + (n_{jr} + n_{sr}) + 1). \quad (5)$$

Since $n_{jr} > 0$, it follows that

$$x_{jr} + x_{sr} \cdot \frac{n_{sr}}{n_{jr}} > \frac{1}{2} \cdot (n_{sr} + 1).$$

Also, $n_{jr} \geq n_{sr}$. Hence

$$x_{jr} + x_{sr} \geq x_{jr} + x_{sr} \cdot \frac{n_{sr}}{n_{jr}} > \frac{1}{2} \cdot (n_{sr} + 1) = \frac{1}{2} \cdot (\min\{n_{jr}, n_{sr}\} + 1).$$

Case 2: v has only a junior child. Then $n_{sr} = x_{sr} = 0$, and

$$\sum_{u \in U_v} \text{dist}(v, u) = x_{jr} \cdot n_{jr} + \sum_{u \in U_{jr}} \text{dist}(v.jr, u). \quad (6)$$

Plugging (2) and (3) in (6), we obtain

$$x_{jr} \cdot n_{jr} = \frac{1}{4} \cdot (2 \cdot n_{jr} + 1).$$

Hence

$$x_{jr} + x_{sr} = x_{jr} > \frac{1}{2} = \frac{1}{2} \cdot (\min\{n_{jr}, n_{sr}\} + 1).$$

■

4.1.2 The Cost Function

In this section, we define and analyze a cost function on abstract binary rooted n -vertex trees. We will show that in order to provide a lower bound for $\text{OPT}(\text{LP2})$, it is sufficient to provide a lower bound for the minimum value of this cost function on a tree from $B_n(h)$.

For a tree $T = (T, rt)$ in $B_n(h)$, let $I = I(T)$ denote the set of inner vertices of T . As in Section 4.1.1, for a vertex v , we define $|v| = |U_v|$, and denote the junior (respectively, senior) child of v by $v.jr$ (resp., $v.sr$). By Lemma 4.4, for any optimal assignment for the values $\{\ell(e) \mid e \in E(T)\}$ of the linear program LP2 holds:

$$\text{OPT}(\text{LP2}) = \sum_{e \in E(T)} \ell(e) = \sum_{v \in I(T)} (\ell(v, v.jr) + \ell(v, v.sr)) \geq \frac{1}{2} \cdot \sum_{v \in I(T)} (\min\{|v.jr|, |v.sr|\} + 1). \quad (7)$$

We call the right-hand side expression the *cost* of T , and denote it $Cost(T)$. In the sequel we provide a lower bound for $\min\{Cost(T) \mid T \in B_n(h)\}$, which would apply to $OPT(LP2)$.

For a tree $T = (T, rt)$ in $B_n(h)$, we call the subtree rooted at the junior (respectively, senior) child of rt the *junior subtree* (resp., *senior subtree*) of T and denote it by $T.jr$ (resp., $T.sr$). Also, let $|T|$ denote the *size* of the tree T , that is, the number of vertices in T .

Consider the following cost function on binary abstract rooted trees:

$$Cost'(T) = Cost'(T.jr) + Cost'(T.sr) + \min\{|T.jr|, |T.sr|\}.$$

It is easy to verify that $Cost'(T)$ can be equivalently expressed as

$$Cost'(T) = \sum_{v \in I(T)} \min\{|v.jr|, |v.sr|\}.$$

By definition, for a tree T in $B_n(h)$, $2 \cdot Cost(T) \geq Cost'(T)$. Hence $\sum_{e \in E(T)} \ell(e) \geq \frac{1}{2} \cdot Cost'(T)$. We will henceforth *focus on proving a lower bound for $Cost'(T)$* , and use the notion “cost” to refer to the function $Cost'$.

Fix a pair of positive integers n and h , $n - 1 \geq h$. Let $R(n, h)$ denote the minimum cost taken over all (n, h) -trees. It follows that

$$Bin(n, h) = MINLA(B_n(h)) \geq \frac{1}{2} \cdot R(n, h). \quad (8)$$

This section is devoted to proving the following theorem, which establishes lower bounds on $R(n, h)$, for all $h \geq \log n$. This theorem is a major component in the proof of our lower bounds.

Theorem 4.5 1. If $\log n \leq h \leq \lfloor 2 \log n \rfloor$, then $R(n, h) \geq \frac{2}{3} \cdot n \cdot \lfloor \frac{1}{8} \log n \rfloor$.

2. For $\lfloor 2 \log n \rfloor < h \leq n - 1$, let $f(h)$ be the minimum integer such that $\binom{h+1}{f(h)} > \frac{2}{3} \cdot n$. Then $R(n, h) > \frac{2}{3} \cdot n \cdot (f(h) - 2)$.

Remark: By Claim A.3, for $h > \lfloor 2 \log n \rfloor$, $\binom{h+1}{\lfloor \log n \rfloor} > \frac{2}{3} \cdot n$, and thus $f(h)$ is well-defined in this range.

Proof: Let n and h be non-negative integers. Consider an (n, h) -tree \tilde{T} . As previously, we assume that for every inner vertex v in \tilde{T} , $|v.jr| \geq |v.sr|$, i.e., that the tree is *senior-adjusted*. It follows that

$$Cost'(\tilde{T}) = \sum_{v \in V(\tilde{T})} |v.sr|.$$

A set of n binary words with at most h bits each will be called an (n, h) -*vocabulary*. Next, we define an injection \mathcal{S} from the set of (n, h) -trees to the set of (n, h) -vocabularies.

For a vertex v in a binary tree T , denote by $P_v = (rt = v_0, v_1, \dots, v_k = v)$ the path from rt to v in T , and define $B_v = b_1 b_2 \dots b_k$ to be its corresponding binary word, where for $i \in \{1, 2, \dots, k\}$, $b_i = 1$ if v_i is the senior child of v_{i-1} , and $b_i = 0$ otherwise. Given an (n, h) -tree T , let $\mathcal{S}(T)$ be the (n, h) -vocabulary that consists of the $|T|$ binary words that correspond to the set of all root-to-vertex paths in T , namely, $\mathcal{S}(T) = \{B_v \mid v \in V(T)\}$. (See Figure 6 for an illustration.)

For a binary word α , denote its *Hamming weight* (the number of 1's in it) by $H(\alpha)$. For a set S of binary words, define its total Hamming weight, henceforth *Hamming cost*, $HCost(S)$, as the sum of Hamming weights of all words in S , namely, $HCost(S) = \sum_{B \in S} H(B)$. Finally, denote the minimum Hamming cost

of a set of n distinct binary words with at most h bits each by $H(n, h)$. Note that the function $H(n, h)$ is monotone non-increasing with h .

The next lemma shows that it is sufficient to prove the desired lower bound for $H(n, h)$.

Lemma 4.6 For all positive integers n and h such that $n - 1 \geq h$, $R(n, h) \geq H(n, h)$.

Proof: For a pair of vertices u and v in a senior-adjusted tree \tilde{T} , define $\chi_v(u)$ to be 1 if u belongs to the subtree rooted at the senior child of v in \tilde{T} , and 0 otherwise. It is easy to verify by a double counting that in a senior-adjusted tree \tilde{T} ,

$$\sum_{v \in V(\tilde{T})} |v.sr| = \sum_{v \in V(\tilde{T})} \sum_{u \in V(\tilde{T})} \chi_v(u) = \sum_{u \in V(\tilde{T})} \sum_{v \in V(\tilde{T})} \chi_v(u) = \sum_{u \in V(\tilde{T})} |\{v \in V(P_u) : u \in U_{v.sr}\}|.$$

Consequently,

$$Cost'(\tilde{T}) = \sum_{v \in V(\tilde{T})} |v.sr| = \sum_{u \in V(\tilde{T})} |\{v \in V(P_u) : u \in U_{v.sr}\}| = \sum_{u \in V(\tilde{T})} H(B_u) = HCost(\mathcal{S}(\tilde{T})).$$

Let T^* be a senior-adjusted (n, h) -tree realizing $R(n, h)$, that is, $Cost'(T^*) = R(n, h)$. It follows that

$$R(n, h) = Cost'(T^*) = HCost(\mathcal{S}(T^*)) \geq H(n, h).$$

■

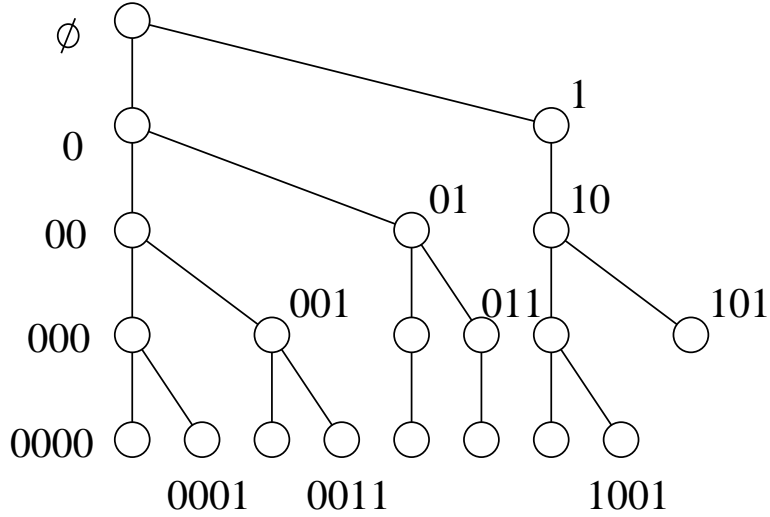


Figure 6: A cost-optimal binary tree for $n = 20$ and $h = 4$. The set of vertex labels of this tree is the set $\mathcal{S}^*(20, 4)$ of 20 words with at most 4 bits each that contains the minimum number of 1's. Obviously, this set contains the 5 words with no 1's at all, that is, the set $\mathcal{S}(4, 0) = \{\emptyset, 0, 00, 000, 0000\}$. In addition, it contains the set $\mathcal{S}(4, 1)$ of 10 words with one single 1, that is, $1, 01, 10, 001, 010, 100, 0001, 0010$, and 1000 . Finally, it also contains some 5 out of 10 words of the set $\mathcal{S}(4, 2)$, specifically, $011, 0011, 101, 0110$, and 1001 .

In what follows we establish lower bounds on $H(n, h)$.

Consider an (n, h) -vocabulary $\mathcal{S}^* = \mathcal{S}^*(n, h)$ realizing $H(n, h)$, that is, a set that satisfies $HCost(\mathcal{S}^*) = H(n, h)$. For a non-negative integer i , $i \leq h$, let $\mathcal{S}(h, i)$ be the set of all distinct binary words with at most h bits each, each containing precisely i 1's. Observe that to contain the minimum total number of 1's, the set \mathcal{S}^* needs to contain all binary words with no 1's, all binary words that contain just a single 1, etc, summing to a total of n binary words. In other words, there exists an integer $r = r(h)$ for which

$$\bigcup_{i=0}^r \mathcal{S}(h, i) \subset \mathcal{S}^* \subseteq \bigcup_{i=0}^{r+1} \mathcal{S}(h, i).$$

(See Figure 6 for an illustration.)

By Fact A.1,

$$|\mathcal{S}(h, i)| = \sum_{k=i}^h \binom{k}{i} = \binom{h+1}{i+1}.$$

Note that for a pair of distinct indices i and j , $0 \leq i, j \leq h$, the sets $\mathcal{S}(h, i)$ and $\mathcal{S}(h, j)$ are disjoint. Since $|\mathcal{S}^*| = n$, it holds that

$$\sum_{i=0}^r \binom{h+1}{i+1} < n \leq \sum_{i=0}^{r+1} \binom{h+1}{i+1}. \quad (9)$$

Recall that for every non-negative integer i , $i \leq h$, each word in $\mathcal{S}(h, i)$ contains precisely i 1's, and thus

$$HCost(\mathcal{S}(h, i)) = i \cdot \binom{h+1}{i+1}.$$

Let

$$N = n - \sum_{i=0}^r \binom{h+1}{i+1} > 0 \quad (10)$$

denote the number of words with Hamming weight $r+1$ in \mathcal{S}^* . Hence

$$H(n, h) = HCost(\mathcal{S}^*) = \sum_{i=0}^r HCost(\mathcal{S}(h, i)) + N \cdot (r+1) = \sum_{i=0}^r i \cdot \binom{h+1}{i+1} + N \cdot (r+1). \quad (11)$$

The next claim establishes a helpful relationship between the parameters h , r , and n .

Claim 4.7 For $h \geq \lfloor 2 \log n \rfloor$, $r \leq \lfloor \frac{h+1}{4} \rfloor - 1$.

Proof: Suppose for contradiction that $r \geq \lfloor \frac{h+1}{4} \rfloor$. Then

$$r+1 \geq \left\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \right\rfloor + 1. \quad (12)$$

By (9), $n \geq 2$. Also, it holds that $h \geq \lfloor 2 \log n \rfloor \geq 2 \lfloor \log n \rfloor$.

We argue that $r < \lfloor \log n \rfloor$. Indeed, otherwise

$$\begin{aligned} \sum_{i=0}^r \binom{h+1}{i+1} &\geq \sum_{i=0}^{\lfloor \log n \rfloor} \binom{h+1}{i+1} \geq \sum_{i=0}^{\lfloor \log n \rfloor} \binom{2 \lfloor \log n \rfloor + 1}{i+1} \\ &> \sum_{i=0}^{\lfloor \log n \rfloor} \binom{\lfloor \log n \rfloor + 1}{i+1} = 2^{\lfloor \log n \rfloor + 1} - 1 \geq n - 1, \end{aligned}$$

contradicting (9). It follows that

$$r+1 \leq \lfloor \log n \rfloor \leq \frac{h}{2}, \quad (13)$$

and consequently,

$$\binom{\lfloor 2 \log n \rfloor + 1}{\left\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \right\rfloor + 1} \leq \binom{h+1}{\left\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \right\rfloor + 1} \leq \binom{h+1}{r+1}.$$

(The last inequality is by (12) and (13).) By Claim A.4,

$$n \leq \binom{\lfloor 2 \log n \rfloor + 1}{\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \rfloor + 1}.$$

Altogether,

$$n \leq \binom{\lfloor 2 \log n \rfloor + 1}{\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \rfloor + 1} \leq \binom{h+1}{r+1} < \sum_{i=0}^r \binom{h+1}{i+1}.$$

However, by (9) the right-hand side is smaller than n , contradiction. \blacksquare

The next lemma establishes lower bounds on $H(n, h)$ for $h \geq \lfloor 2 \log n \rfloor$. Since $H(n, h)$ is monotone non-increasing with h , it follows that the lower bound for $h = \lfloor 2 \log n \rfloor$ also applies for all smaller values of h . The statement of this lemma is very similar to the statement of Theorem 4.5, except that it establishes bounds for $H(n, h)$ while Theorem 4.5 does so for $R(n, h)$. By Lemma 4.6, $R(n, h) \geq H(n, h)$, and thus this lemma is the last step in the proof of Theorem 4.5.

Lemma 4.8 1. $H(n, \lfloor 2 \log n \rfloor) \geq \frac{2}{3} \cdot n \cdot \lfloor \frac{1}{8} \log n \rfloor$.

2. For any $\lfloor 2 \log n \rfloor < h \leq n - 1$, $H(n, h) > \frac{2}{3} \cdot n \cdot (f(h) - 2)$. (See the statement of Theorem 4.5 for the definition of $f(h)$.)

Proof: By Lemma A.2,

$$\sum_{i=0}^r \binom{h+1}{i+1} < \frac{3}{2} \cdot \binom{h+1}{r+1}.$$

By (10),

$$\sum_{i=0}^r \binom{h+1}{i+1} = n - N.$$

It follows that $\binom{h+1}{r+1} > \frac{2}{3} \cdot (n - N)$, and so, $\binom{h+1}{r+1} + N > \frac{2}{3} \cdot n$. Hence, by (11),

$$H(n, h) = \sum_{i=0}^r i \cdot \binom{h+1}{i+1} + N \cdot (r+1) > r \cdot \binom{h+1}{r+1} + N \cdot (r+1) > \frac{2}{3} \cdot n \cdot r. \quad (14)$$

The analysis splits into two cases.

Case 1: $h = \lfloor 2 \log n \rfloor$.

We will prove the first assertion of Lemma 4.8, that is, $H(n, \lfloor 2 \log n \rfloor) \geq \frac{2}{3} \cdot n \cdot \lfloor \frac{1}{8} \log n \rfloor$. We assume that $n \geq 256$, as otherwise the right-hand side vanishes and the statement holds trivially.

The next claim shows that in this case r is quite large.

Claim 4.9 $r \geq \lfloor \frac{1}{8} \log n \rfloor$.

Proof: Suppose for contradiction that $r \leq \lfloor \frac{1}{8} \log n \rfloor - 1$. Observe that for $n \geq 3$, $\lfloor \frac{1}{8} \log n \rfloor + 1 \leq \lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \rfloor$, and thus Lemma A.2 is applicable. Hence,

$$\sum_{i=0}^{\lfloor \frac{1}{8} \log n \rfloor + 1} \binom{\lfloor 2 \log n \rfloor + 1}{i} < \frac{3}{2} \cdot \binom{\lfloor 2 \log n \rfloor + 1}{\lfloor \frac{1}{8} \log n \rfloor + 1}. \quad (15)$$

By (9),

$$n \leq \sum_{i=0}^{r+1} \binom{h+1}{i+1} = \sum_{i=0}^{r+1} \binom{\lfloor 2 \log n \rfloor + 1}{i+1} \leq \sum_{i=0}^{\lfloor \frac{1}{8} \log n \rfloor} \binom{\lfloor 2 \log n \rfloor + 1}{i+1} = \sum_{i=0}^{\lfloor \frac{1}{8} \log n \rfloor + 1} \binom{\lfloor 2 \log n \rfloor + 1}{i} - 1.$$

However, (15) implies that the right-hand side is strictly smaller than $\frac{3}{2} \cdot \binom{\lfloor 2 \log n \rfloor + 1}{\lfloor \frac{1}{8} \log n \rfloor + 1} - 1 \leq n$, contradiction. ■

Consequently, by (14), we have

$$H(n, \lfloor 2 \log n \rfloor) > \frac{2}{3} \cdot n \cdot r \geq \frac{2}{3} \cdot n \cdot \left\lfloor \frac{1}{8} \log n \right\rfloor.$$

This completes the proof of the first assertion of Lemma 4.8.

To prove the second assertion we analyze the case $h > \lfloor 2 \log n \rfloor$.

Case 2: $h > \lfloor 2 \log n \rfloor$.

We start the analysis of this case by showing that for h in this range, the upper bound on the value of r established in Claim 4.7 can be slightly improved.

Claim 4.10 $r \leq \lfloor \frac{h+1}{4} \rfloor - 2$.

Proof: It is easy to verify that $n \leq \binom{\lfloor 2 \log n \rfloor + 2}{\lfloor \frac{\lfloor 2 \log n \rfloor + 2}{4} \rfloor}$. Suppose for contradiction that $r \geq \lfloor \frac{h+1}{4} \rfloor - 1$. Then $r+1 \geq \lfloor \frac{\lfloor 2 \log n \rfloor + 2}{4} \rfloor$. Hence

$$n \leq \binom{\lfloor 2 \log n \rfloor + 2}{\lfloor \frac{\lfloor 2 \log n \rfloor + 2}{4} \rfloor} \leq \binom{h+1}{r+1} < \sum_{i=0}^r \binom{h+1}{i+1}.$$

However, by (9), the right-hand side is smaller than n , contradiction. ■

To complete the proof, we provide a lower bound on r .

Claim 4.11 $\binom{h+1}{r+2} > \frac{2}{3} \cdot n$.

Proof: By Claim 4.10, $r \leq \lfloor \frac{h+1}{4} \rfloor - 2$. Hence, Lemma A.2 implies that

$$\sum_{i=0}^{r+1} \binom{h+1}{i+1} < \frac{3}{2} \cdot \binom{h+1}{r+2}.$$

Consequently, by (9),

$$n \leq \sum_{i=0}^{r+1} \binom{h+1}{i+1} < \frac{3}{2} \cdot \binom{h+1}{r+2}.$$

■

Recall that $f(h)$ is defined to be the minimum integer that satisfies $\binom{h+1}{f(h)} > \frac{2}{3} \cdot n$. Claim 4.11 implies that $r \geq f(h) - 2$. By (14), it follows that

$$H(n, h) > \frac{2}{3} \cdot n \cdot r \geq \frac{2}{3} \cdot n \cdot (f(h) - 2),$$

implying the second assertion of Lemma 4.8. ■

Lemmas 4.6 and 4.8 imply Theorem 4.5. ■

We are now ready to derive the desired lower bound for binary trees.

Theorem 4.12 For a sufficiently large integer n , and h , $h \geq \log n$, the minimum weight $\text{Bin}(n, h)$ of a ϑ -tree in $B_n(h)$ is at least $\Omega(\Psi \cdot n)$, for some Ψ satisfying $h = \Omega(\Psi \cdot n^{1/\Psi})$.

Proof: First, by (8), $\text{Bin}(n, h) = \text{MINLA}(B_n(h)) \geq \frac{1}{2} \cdot R(n, h)$. The analysis splits into three cases, depending on the value of h .

Case 1: $\log n \leq h \leq \lfloor 2 \log n \rfloor$. By the first assertion of Theorem 4.5,

$$\text{Bin}(n, h) \geq \frac{1}{2} \cdot R(n, h) = \Omega(\log n \cdot n).$$

Observe that for $h \in [\log n, \lfloor 2 \log n \rfloor]$, and $\Psi = \log n$, it holds that $h = \Omega(\Psi \cdot n^{1/\Psi})$, and we are done.

Case 2: $\lfloor 2 \log n \rfloor \leq h \leq n^{1/4}$. By the second assertion of Theorem 4.5,

$$\text{Bin}(n, h) \geq \frac{1}{2} \cdot R(n, h) \geq \frac{1}{3} \cdot n \cdot (f(h) - 2),$$

where $f(h)$ is the minimum integer that satisfies $\binom{h+1}{f(h)} > \frac{2}{3} \cdot n$. Observe that for a sufficiently large n , and $h \in [\lfloor 2 \log n \rfloor, n^{1/4}]$, we have that $f(h) \geq 4$, and so, $f(h) - 2 \geq \frac{1}{2} \cdot f(h)$. It follows that $\text{Bin}(n, h) = \Omega(f(h) \cdot n)$. Also, it holds that

$$\left(\frac{e \cdot (h+1)}{f(h)} \right)^{f(h)} > \binom{h+1}{f(h)} > \frac{2}{3} \cdot n.$$

Consequently,

$$h > \left(\frac{1}{e} \cdot f(h) \cdot \left(\frac{2}{3} \cdot n \right)^{\frac{1}{f(h)}} \right) - 1 = \Omega\left(f(h) \cdot n^{\frac{1}{f(h)}}\right).$$

Case 3: $h > n^{1/4}$. In this range any constant $\Psi \geq 4$ satisfies $h = \Omega(\Psi \cdot n^{1/\Psi})$. Note that the weight of the minimum spanning tree of ϑ is $n - 1$, which implies that for a constant Ψ ,

$$\text{Bin}(n, h) \geq n - 1 = \Omega(\Psi \cdot n).$$

■

4.1.3 Lower Bounds for General High Trees

In this section we show that our lower bound for binary ϑ -trees (Theorem 4.12) implies an analogous lower bound for general *high* ϑ -trees, that is, ϑ -trees of depth $h \geq \log n$. We do this in two stages. First, we show that a lower bound for 4-ary trees will suffice. Second, we show that the lower bound for binary trees implies the desired lower bound for 4-ary trees.

For an inner node v in a tree T , let $ch(v)$ denote the number of its children in T . We order the children $c_1(v), c_2(v), \dots, c_{ch(v)}(v)$ of v so that the sizes of their corresponding subtrees form a monotone non-increasing sequence, i.e., $|T_{c_1(v)}| \geq |T_{c_2(v)}| \geq \dots \geq |T_{c_{ch(v)}(v)}|$. (Observe that $c_1(v), c_2(v), \dots, c_{ch(v)}(v)$ are vertices of a ϑ -tree, and thus, they are all numbers from the set $[n]$. However, it does not necessarily hold that $c_1(v) < c_2(v) < \dots < c_{ch(v)}(v)$.)

For a vertex v in T , we define its *star subgraph* S_v as the subgraph of T connecting v to its children in T , namely, $S_v = (V_v, E_v)$, where $V_v = \{v, c_1(v), \dots, c_{ch(v)}(v)\}$ and $E_v = \{(v, c_i(v)) \mid i = 1, 2, \dots, ch(v)\}$. For convenience, we denote $T_i = T_{c_i(v)}$, for $i = 1, 2, \dots, ch(v)$.

The Procedure *Full* accepts as input a star subgraph S_v of a tree T and transforms it into a full binary tree \mathcal{B}_v rooted at v that has depth $\lfloor \log(ch(v) + 1) \rfloor$, such that $c_1(v), c_2(v), \dots, c_{ch(v)}$ are arranged in the

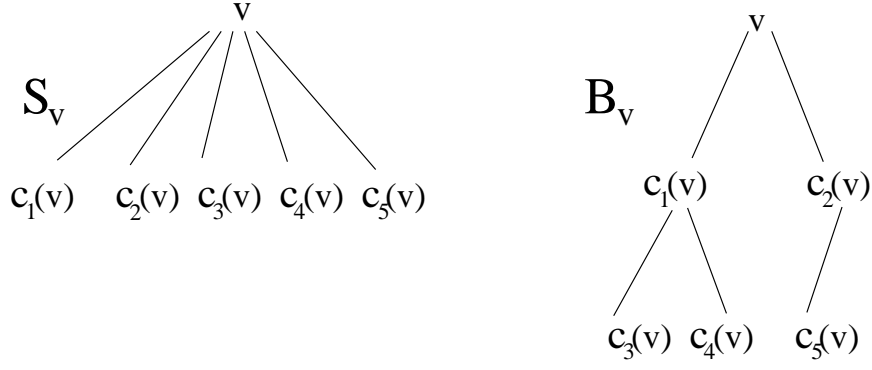


Figure 7: The tree S_v on the left is the star subgraph rooted at v , and having the five vertices $c_1(v), c_2(v), \dots, c_5(v)$ as its leaves. The full binary tree B_v on the right is obtained as a result of the invocation of the Procedure *Full* on S_v .

resulting tree by increasing order of level. (See Figure 7 for an illustration.) Specifically, $c_1(v)$ and $c_2(v)$ become the two children of v in B_v , respectively. More generally, for each index $i = 1, 2, \dots, \lfloor \frac{ch(v)-1}{2} \rfloor$, $c_{2i+1}(v)$ and $c_{2i+2}(v)$ become the two children of the vertex $c_i(v)$, respectively.

The Procedure *4Extension* accepts as input a tree T and transforms it into a 4-ary tree spanning the original set of vertices. Basically, the procedure invokes the Procedure *Full* on every star of the original tree T . More specifically, if the tree T contains only one node, then the Procedure *4Extension* leaves the tree intact. Otherwise, it is invoked recursively on each of the subtrees $T_1, T_2, \dots, T_{ch(rt)}$ of T . Once the Procedure *4Extension* is done with the recursive invocations, the Procedure *Full* is invoked with the parameter S_{rt} and transforms the star subgraph S_{rt} of the root into a full binary tree B_{rt} as described above. (See Figure 8 for an illustration.)

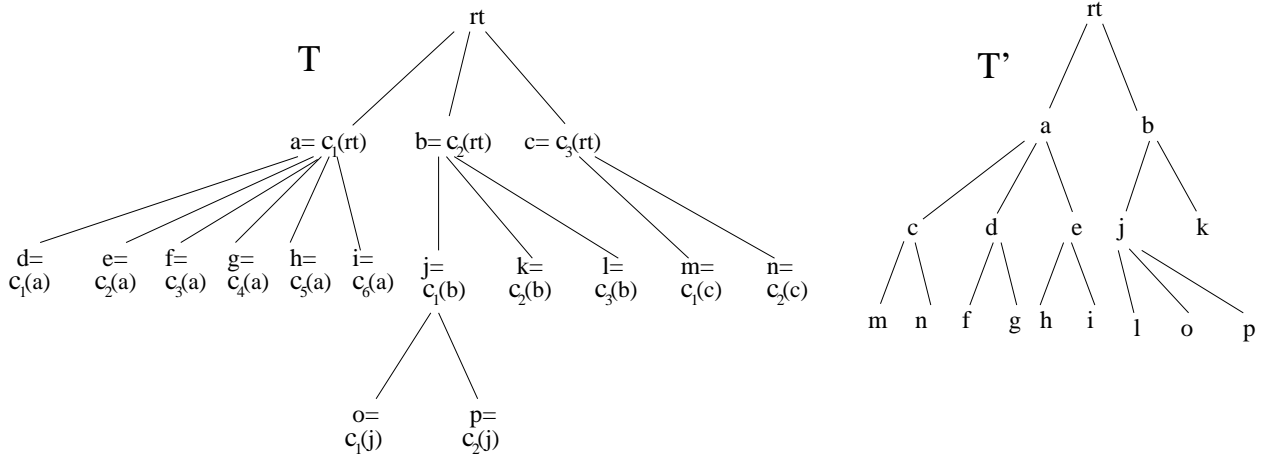


Figure 8: A spanning tree T of $\{rt, a, b, \dots, p\}$ rooted at the vertex rt is depicted on the left. The 4-ary spanning tree T' of $\{rt, a, b, \dots, p\}$ rooted at rt is depicted on the right. The tree T' is obtained as a result of the invocation of the Procedure *4Extension* on T .

It is easy to verify that for any tree T , the tree T' obtained as a result of the invocation of the Procedure *4Extension* on the tree T has the same vertex set. Moreover, in the following lemma we show that no vertex in T' has more than four children.

Lemma 4.13 T' is a 4-ary tree such that its root rt has at most two children.

Proof: The vertices $c_1(rt)$ and $c_2(rt)$ are the only children of rt in T' . For a vertex $v \in V$, $v \neq rt$, let u denote the parent of v in the original tree T . Let i be the index such that $v = c_i(u)$. If $i \in \{1, 2\}$, then u remains the parent of v in T' . Otherwise, $c_{\lfloor (i-1)/2 \rfloor}(u)$ becomes the parent of v in T' . Moreover, v will have at most four children in T' , specifically, $c_{2i+1}(u)$, $c_{2i+2}(u)$, $c_1(v)$ and $c_2(v)$. If v is a leaf in T then it will have at most two children in T' , $c_{2i+1}(u)$ and $c_{2i+2}(u)$. If $i > \lfloor \frac{ch(u)-1}{2} \rfloor$, then v may have only two children in T' , $c_1(v)$ and $c_2(v)$. In this case if v is a leaf in T then it is a leaf in T' as well. ■

Remark: Observe that when this procedure is invoked on a binary tree, it does not change it and returns $T' = T$. Also, for a ternary tree, the resulting tree T' is ternary as well.

In the next lemma we show that the depth $h(T')$ of T' is not much greater than that of T .

Lemma 4.14 $h(T') \leq h(T) + \log |T|$.

Proof: The proof is by induction on $h = h(T)$.

Base: $h = 0$. The claim holds vacuously in this case, since $T' = T$.

Induction Step: We assume the correctness of the claim for all trees of depth at most $h - 1$, and prove it for trees of depth h .

Consider a tree T of depth h . By the induction hypothesis, for all indices i , $1 \leq i \leq ch(rt)$,

$$h(T'_i) \leq h(T_i) + \log |T_i|. \quad (16)$$

Observe that the level of $c_i(rt)$ in T' is $\lfloor \log(i + 1) \rfloor$. Hence the depth $h(T')$ of T' is given by

$$h(T') = \max\{h(T'_i) + \lfloor \log(i + 1) \rfloor : i \in \{1, 2, \dots, ch(rt)\}\}. \quad (17)$$

Let t be an index in $\{1, 2, \dots, ch(rt)\}$ realizing the maximum, i.e., $h(T') = h(T'_t) + \lfloor \log(t + 1) \rfloor$. By (16) and (17),

$$h(T') \leq h(T_t) + \log |T_t| + \lfloor \log(t + 1) \rfloor.$$

Note also that $h(T_t) \leq h - 1$, and $\lfloor \log(t + 1) \rfloor \leq \lfloor \log t \rfloor + 1$. Hence, it holds that

$$h(T') \leq h + \log |T_t| + \lfloor \log t \rfloor.$$

Recall that the children of rt are ordered such that $|T_1| \geq |T_2| \geq \dots \geq |T_{ch(rt)}|$.

Hence, $t \cdot |T_t| \leq \sum_{i=1}^t |T_i| \leq |T|$. Consequently, $\log |T_t| + \lfloor \log t \rfloor \leq \log |T|$, and we are done. ■

In the following lemma we argue that the weight of the resulting tree T' is not much greater than the weight of the original tree T .

Lemma 4.15 $w(T') \leq 3 \cdot w(T)$.

Proof: For each vertex v in T , its star subgraph $S_v = (V_v, E_v)$ is replaced by a full binary tree $F_v = Full(S_v) = (V_v, E'_v)$. The weight of S_v is given by

$$w(S_v) = \sum_{i=1}^{ch(v)} w(v, c_i(v)).$$

Next, we show that the weight of F_v is at most three times greater than the weight of S_v . This will imply the statement of the lemma.

By the triangle inequality, for each edge $e = (x, y)$ in E'_v , we have $w(e) \leq w(v, x) + w(v, y)$. Therefore,

$$w(F_v) = \sum_{e \in E'_v} w(e) \leq \sum_{e=(x,y) \in E'_v} (w(v, x) + w(v, y)) .$$

Denote the degree of a vertex z in $F_v = (V, E'_v)$ by $\deg(z, F_v)$. It is easy to verify by double counting that

$$\sum_{e=(x,y) \in E'_v} (w(v, x) + w(v, y)) = \sum_{u \in V_v} \deg(u, F_v) \cdot w(v, u) .$$

Since F_v is a binary tree, for each vertex z in F_v , $\deg(z, F_v) \leq 3$. Hence,

$$w(F_v) \leq 3 \cdot \sum_{u \in V_v} w(v, u) = 3 \cdot \sum_{i=1}^{ch(v)} w(v, c_i(v)) + 3 \cdot w(v, v) = 3 \cdot w(S_v).$$

■

The next corollary summarizes the properties of the Procedure *4Extension*.

Corollary 4.16 *For a ϑ -tree $T = (V, E)$, the Procedure *4Extension*(T) constructs a 4-ary ϑ -tree $T' = (V, E')$ such that $h(T) \leq h(T') \leq h(T) + \log |T|$ and $w(T') \leq 3 \cdot w(T)$.*

Next, we demonstrate that a 4-ary tree can be transformed into a binary tree, while increasing the depth and weight only by constant factors. (We did not make any special effort to minimize these constant factors.)

The Procedure *Path* accepts as input a star subgraph S_v of a tree T and transforms it into a path $P_v = (c_0(v) = v, c_1(v), \dots, c_{ch(v)}(v))$. The notation $c_i(v)$ refers to the i th child of v in the tree T' that is provided to the Procedure *Path* as *input*. (See Figure 9 for an illustration.)

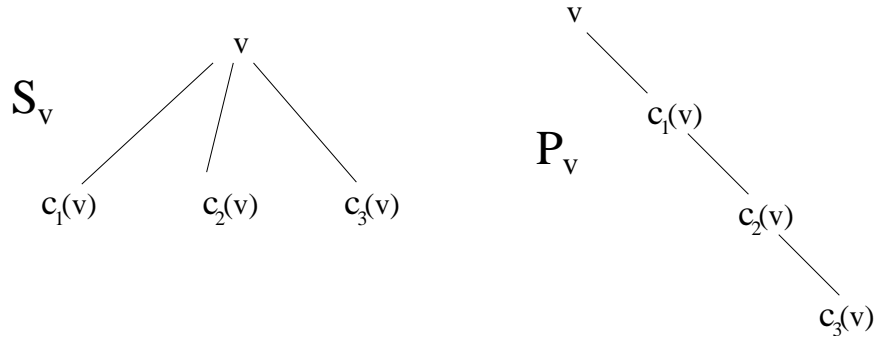


Figure 9: The tree S_v on the left is the star subgraph rooted at v and having the three vertices $c_1(v), c_2(v), c_3(v)$ as its leaves. The path P_v on the right is obtained as a result of the invocation of the Procedure *Path* on S_v .

The Procedure *BinExtension* accepts as input a 4-ary tree $T' = (V, E')$ rooted at a given vertex $rt \in V$ and transforms it into a binary tree spanning the original set of vertices. If the tree T' contains only one node, then the Procedure *BinExtension* leaves the tree intact. Otherwise, for every inner vertex $v \in V$, the procedure replaces the star $S_v = (v, c_1(v), \dots, c_{ch(v)})$ by the path $P_v = \text{Path}(S_v)$. (See Figure 10 for an illustration.)

In the next lemma we argue that $T'' = \text{BinExtension}(T')$ is a binary tree.

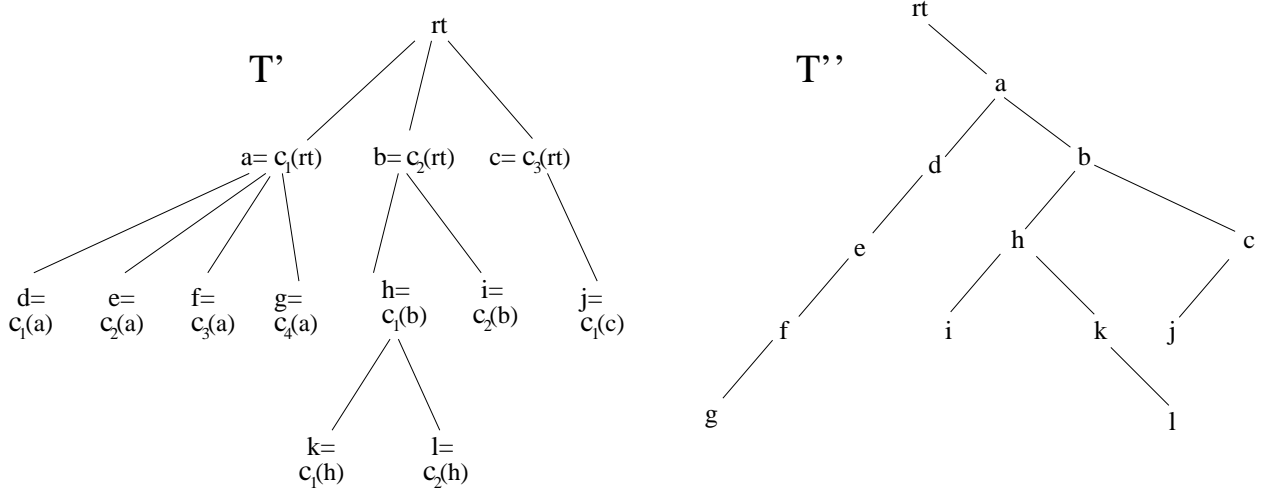


Figure 10: A 4-ary spanning tree T' of $\{rt, a, b, \dots, l\}$ rooted at the vertex rt is depicted on the left. The binary spanning tree T'' of $\{rt, a, b, \dots, l\}$ rooted at rt is depicted on the right. The tree T'' is obtained as a result of the invocation of the Procedure *BinExtension* on T' .

Lemma 4.17 T'' is a binary tree. Moreover, its root rt has at most one child.

Proof: The vertex $c_1(rt)$ is the only child of rt in T' . For a vertex $v \in V$, $v \neq rt$, let u denote the parent of v in the tree T' . Let i be the index such that $v = c_i(u)$. If $i = 1$ then u is the parent of v in T'' . Otherwise, $c_{i-1}(u)$ is the parent of v in T'' . Moreover, if $i < ch(u)$ then v may have two children in T'' , specifically, $c_{i+1}(u)$ and $c_1(v)$. If v is a leaf in T' then it will have only one child, $c_{i+1}(u)$. If $i = ch(u)$ then v may have at most one child in T'' , specifically, $c_1(v)$. In this case if v is a leaf in T' then it is a leaf in T'' as well. ■

The next two statements imply that both the depth and the weight of T' and T'' are the same, up to a constant factor.

Lemma 4.18 $h(T') \leq h(T'') \leq 4 \cdot h(T')$.

Proof: For each vertex v in T' , its star subgraph S_v is transformed into a path $Path(S_v)$ of depth $ch(v)$. Since T' is a 4-ary tree, $ch(v) \leq 4$. Thus the ratio between the depth of $Path(S_v)$ and the depth of S_v is at most 4. It follows that for every vertex $v \in V$, the distance between the root and v in T'' is at most four times larger than the distance between them in T' . ■

Lemma 4.19 $w(T'') \leq 2 \cdot w(T')$.

The proof of this lemma is very similar to that of Lemma 4.15, and it is omitted.

We summarize the properties of the reduction from 4-ary to binary trees in the following corollary.

Corollary 4.20 For a 4-ary ϑ -tree $T' = (V, E')$, the Procedure *BinExtension*(T') constructs a binary ϑ -tree T'' such that $h(T') \leq h(T'') \leq 4 \cdot h(T')$ and $w(T'') \leq 2 \cdot w(T')$.

Corollary 4.16 and Corollary 4.20 imply the following statement.

Lemma 4.21 For a high ϑ -tree $T = (V, E)$, the invocation *BinExtension*(*4Extension*(T)) returns a binary ϑ -tree T'' such that $h(T) \leq h(T'') \leq 8 \cdot h(T)$ and $w(T'') \leq 6 \cdot w(T)$.

We are now ready to prove the desired lower bound for high ϑ -trees.

Theorem 4.22 *For sufficiently large integers n and h , $h \geq \log n$, the minimum weight $W(n, h)$ of a general ϑ -tree that has depth at most h is at least $\Omega(\Psi \cdot n)$, for some Ψ satisfying $h = \Omega(\Psi \cdot n^{1/\Psi})$.*

Proof: Given an arbitrary ϑ -tree T of depth $h(T)$ at least $\log n$, Lemma 4.21 implies the existence of a binary tree \tilde{T} such that $h(\tilde{T}) \leq 8 \cdot h(T)$ and $w(\tilde{T}) \leq 6 \cdot w(T)$. By Theorem 4.12, the weight $w(\tilde{T})$ of \tilde{T} is at least $\Omega(\Psi \cdot n)$, for some Ψ satisfying $h(\tilde{T}) = \Omega(\Psi \cdot n^{1/\Psi})$. Since $w(T) = \Omega(w(\tilde{T}))$ and $h(T) = \Omega(h(\tilde{T}))$, it follows that the weight $w(T)$ of T is at least $\Omega(\Psi \cdot n)$ as well, and Ψ satisfies the required relation $h(T) = \Omega(h(\tilde{T})) = \Omega(\Psi \cdot n^{1/\Psi})$. ■

Clearly, the lightness of a graph G is at least as large as that of any BFS tree of G , implying the following result.

Corollary 4.23 *For sufficiently large integers n and h , $h \geq \log n$, any spanning subgraph with hop-radius at most h has lightness at least $\Omega(\Psi)$, for some Ψ satisfying $h = \Omega(\Psi \cdot n^{1/\Psi})$.*

4.2 Lower Bounds for Low Trees

In this section we devise lower bounds for the weight of *low* ϑ -trees, that is, ϑ -trees of depth $h < \log n$.

Our strategy is to show lower bounds for the covering (Section 4.2.1), and then to translate them into lower bounds for the weight (Section 4.2.2).

4.2.1 Lower Bounds for Covering

In this section we prove lower bounds for the covering of ϑ -trees that have depth $h \leq \frac{1}{5} \cdot \log n$. (See Section 2 for the definition of covering and Section 3 for some of its properties.)

A spanning tree T of an n -point 1-dimensional Euclidean space M is called a *flat tree*. Note that the notion of flat tree generalizes the notion of ϑ -tree. Also, all notions defined in Section 2 for ϑ -trees (e.g., covering, left and right children) are well-defined for flat trees. It turns out that some properties of ϑ -trees are easier to derive for the more general flat trees. For this reason in this section we prove a number of statements for flat trees. These statements will be later used for ϑ -trees.

The following lemma shows that for any flat tree T , the covering of T cannot be much smaller than the maximum degree of a vertex in T .

Lemma 4.24 *For a flat tree T and a vertex v in T , the covering $\gamma(T)$ of T is at least $(\deg(v) - 2)/2$.*

Proof: Consider an edge (v, u) in T . Since every edge adjacent to v is either left or right with respect to v , by the pigeonhole principle either at least $\left\lceil \frac{\deg(v)}{2} \right\rceil$ edges are left with respect to v or at least $\left\lceil \frac{\deg(v)}{2} \right\rceil$ edges are right with respect to v . Suppose without loss of generality that at least $\left\lceil \frac{\deg(v)}{2} \right\rceil$ edges are right with respect to v , and denote by \mathcal{R} the set of edges which are adjacent to v and right with respect to v . Note that all these edges except possibly the edge $(v, v+1)$ cover the vertex $(v+1)$, and thus the covering of $(v+1)$ is at least

$$|\mathcal{R}| - 1 \geq \left\lceil \frac{\deg(v)}{2} \right\rceil - 1 \geq \frac{\deg(v) - 2}{2}.$$

■

Though the ultimate goal of this section is to establish a lower bound for the range $h \leq \frac{1}{5} \cdot \log n$, our argument provides a relationship between covering and depth h in a wider range $h \leq \ln n$. The next lemma takes care of the simple case of h close to $\ln n$. The far more complicated case when h is small is analyzed in Lemma 4.26.

Lemma 4.25 For a flat tree that has depth h and covering γ , such that $\frac{\ln n}{2} < h \leq \ln n$,

$$\gamma > \frac{1}{e^2} \cdot h \cdot n^{1/h} - h.$$

Proof: The right hand-side is negative for h in this range. The lemma follows since $\gamma \geq 0$. \blacksquare

Lemma 4.26 For a flat tree that has depth h and covering γ , such that $h \leq \frac{\ln n}{2}$,

$$\gamma > \frac{1}{10} \cdot h \cdot n^{1/h} - h.$$

Proof:

The proof is by induction on h , for all values of $n \geq e^{2h}$.

Base: The statement holds vacuously for $h = 0$. For $h = 1$, the degree of the root is $n - 1$. Hence Lemma 4.24 implies that the covering is at least $\frac{n-3}{2} \geq \frac{1}{10} \cdot n - 1$, for any $n \geq e^2$.

Induction Step: We assume the claim for all flat trees of depth at most h , and prove it for flat trees of depth $h + 1$. Let T be a flat tree on n vertices that has depth $h + 1$ and covering γ , such that $h + 1 \leq \frac{\ln n}{2}$. If $\deg(rt) \geq \sqrt{n}$, then Lemma 4.24 implies that the covering is at least $\frac{\sqrt{n}-2}{2}$. It is easy to verify that for $h \geq 1$ and $n \geq e^{2(h+1)}$,

$$\frac{\sqrt{n}-2}{2} \geq \frac{1}{10} \cdot (h+1) \cdot n^{1/(h+1)} - (h+1).$$

Henceforth, we assume that $\deg(rt) < \sqrt{n}$.

For a child u of rt , denote by T_u the subtree of T rooted at u . Such a subtree T_u will be called *light* if $|T_u| < e^h$. Denote the set of all light subtrees of T by \mathcal{L} , and define T' as the tree obtained from T by omitting all light subtrees from it, namely, $T' = T \setminus \bigcup_{T_u \in \mathcal{L}} T_u$. Observe that the covering γ' of T' is at

most γ . Next, we obtain a lower bound on γ' .

Observe that

$$\left| \bigcup_{T_u \in \mathcal{L}} T_u \right| = \sum_{T_u \in \mathcal{L}} |T_u| < \deg(rt) \cdot e^h < \sqrt{n} \cdot \frac{\sqrt{n}}{e} = \frac{n}{e},$$

implying that

$$n' = |T'| = |T| - \left| \bigcup_{T_u \in \mathcal{L}} T_u \right| > n - n/e. \quad (18)$$

Denote the subtrees of T' by T_1, \dots, T_k , where $k \leq \deg(rt)$, and define for each $1 \leq i \leq k$, $n_i = |T_i|$, $\gamma_i = \gamma(T_i)$, and $h_i = \text{depth}(T_i)$. Fix an index i , $i \in \{1, \dots, k\}$. Observe that $n_i \geq e^h$, or equivalently, $h \leq \ln n_i$. Since $h_i \leq h$, we have that $h_i \leq \ln n_i$. By the induction hypothesis and Lemma 4.25,

$$\gamma_i > \frac{1}{10} \cdot h_i \cdot n_i^{1/h_i} - h_i.$$

As the function $f(x) = \frac{1}{10} \cdot x \cdot n_i^{1/x} - x$ is monotone decreasing in the range $[1, \ln n_i]$, and since $h_i \leq h \leq \ln n_i$, it follows that

$$\gamma_i > \frac{1}{10} \cdot h_i \cdot n_i^{1/h_i} - h_i \geq \frac{1}{10} \cdot h \cdot n_i^{1/h} - h.$$

Therefore, for each index i , $1 \leq i \leq k$,

$$n_i < \left(\frac{\gamma_i + h}{\frac{1}{10} \cdot h} \right)^h. \quad (19)$$

For each i , $1 \leq i \leq k$, choose some vertex v_i^* in T_i with maximum covering, i.e., $\gamma_{T_i}(v_i^*) = \gamma_i$. We assume without loss of generality that

$$v_1^* < v_2^* < \dots < v_k^*.$$

(Note that this order is not necessarily the order of the children u_1, u_2, \dots, u_k of the root rt of T' . In other words, it may be the case that $v_1^* < v_2^*$ but $u_1 > u_2$.) Let p be the index for which $v_p^* < rt$ and $v_{p+1}^* > rt$. (See Figure 11 for an illustration.)

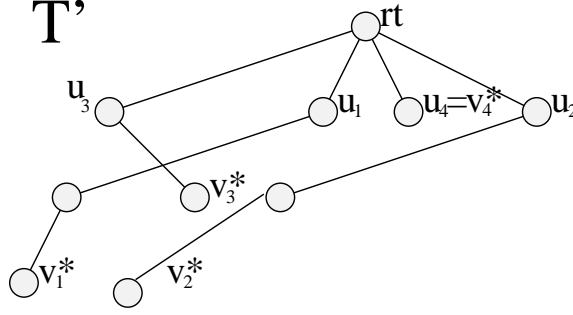


Figure 11: The tree T' is a flat tree rooted at rt . The child u_i of rt , for each $1 \leq i \leq 4$, is the root of the non-light subtree T_i . The vertex in T_i with maximum covering is denoted by v_i^* , with $v_1^* < v_2^* < v_3^* < v_4^*$. (Vertices in T_i which do not belong to the path connecting u_i and v_i^* do not appear in the figure.) The vertex v_3^* must be covered by at least two distinct edges: one that belongs to the path connecting rt and v_1^* , and another one that belongs to the path connecting rt and v_2^* ; it is also covered by the edge (rt, u_3) , and so its covering is 3.

Claim 4.27 $\gamma' \geq \max\{\gamma_1, \gamma_2 + 1, \dots, \gamma_p + (p - 1)\}$.

Proof: Consider the path in T' between rt and v_i^* , for each $1 \leq i \leq p - 1$. For each index j , $i + 1 \leq j \leq p$, the vertex v_j^* must be covered by at least one edge in that path. It follows that $\gamma_{T'}(v_j^*) \geq \gamma_j + j - 1$, for each $1 \leq j \leq p$, as required. ■

A symmetric argument yields the following inequality.

Claim 4.28 $\gamma' \geq \max\{\gamma_k, \gamma_{k-1} + 1, \dots, \gamma_{p+1} + (k - p - 1)\}$.

Suppose without loss of generality that $\sum_{i=1}^p n_i \geq \sum_{i=p+1}^k n_i$. (The argument is symmetric if this is not the case.) Since $n' = |T'| = \sum_{i=1}^k n_i$, it follows that $\frac{n'}{2} \leq \sum_{i=1}^p n_i$. By (19), $\sum_{i=1}^p n_i < \sum_{i=1}^p \left(\frac{\gamma_i + h}{\frac{1}{10} \cdot h}\right)^h$. Notice that for each $1 \leq i \leq p$, $\gamma_i + i - 1 \leq \gamma'$. Consequently,

$$\sum_{i=1}^p \left(\frac{\gamma_i + h}{\frac{1}{10} \cdot h}\right)^h \leq \sum_{i=1}^p \left(\frac{\gamma' + h - (i - 1)}{\frac{1}{10} \cdot h}\right)^h = 10^h \cdot \left(\frac{1}{h}\right)^h \cdot \sum_{i=1}^p (\gamma' + h - (i - 1))^h.$$

Since $\sum_{i=1}^p (\gamma' + h - (i-1))^h < \frac{(\gamma'+h+1)^{h+1}}{h+1}$, we conclude that

$$\begin{aligned} \frac{n'}{2} &< 10^h \cdot \left(\frac{1}{h}\right)^h \cdot \frac{(\gamma' + h + 1)^{h+1}}{h + 1} \\ &= \frac{1}{10} \cdot \left(\frac{h+1}{h}\right)^h \cdot \left(\frac{\gamma' + h + 1}{\frac{1}{10} \cdot (h+1)}\right)^{h+1} \leq \frac{e}{10} \cdot \left(\frac{\gamma' + h + 1}{\frac{1}{10} \cdot (h+1)}\right)^{h+1}. \end{aligned} \quad (20)$$

By (18), $n - n/e < n'$, and thus (20) implies that

$$\frac{n \cdot (1 - 1/e)}{2} < \frac{e}{10} \cdot \left(\frac{\gamma' + h + 1}{\frac{1}{10} \cdot (h+1)}\right)^{h+1},$$

and consequently, $n < \left(\frac{\gamma'+h+1}{\frac{1}{10} \cdot (h+1)}\right)^{h+1}$. It follows that $\gamma \geq \gamma' > \frac{1}{10} \cdot (h+1) \cdot n^{\frac{1}{h+1}} - (h+1)$, as required. \blacksquare

By Lemmas 4.25 and 4.26, for $h \leq \ln n$,

$$\gamma > \frac{1}{10} \cdot h \cdot n^{1/h} - h. \quad (21)$$

The next corollary follows easily from Lemma 4.26. Its objective is to get rid of the minus h in the right-hand side of (21).

Corollary 4.29 *For a flat tree that has depth h and covering γ , such that $h \leq \frac{1}{5} \cdot \log n$, it holds that $\gamma > \frac{1}{20} \cdot h \cdot n^{1/h}$.*

Proof: It is easy to verify that for any $h \leq \frac{1}{5} \cdot \log n$, it holds that $h < \frac{1}{2} \cdot \left(\frac{1}{10} \cdot h \cdot n^{1/h}\right)$. Hence the statement follows from Lemma 4.26. \blacksquare

4.2.2 Lower Bounds for Weight

In this section we employ the lower bound for the covering (Corollary 4.29) to show lower bounds for the weight of ϑ -trees of depth $h < \log n$. For the range $h \leq \frac{1}{10} \cdot \log n$, we employ a technique due to Agarwal et al. [2] to translate the lower bound on the covering established in the previous section into the desired lower bound for low trees. (Our lower bound on the covering is, however, significantly stronger than that of [2].) Somewhat surprisingly, our lower bound for the complementary range $\frac{1}{10} \cdot \log n < h < \log n$ relies on our lower bound for the range $h \geq \log n$ (Theorem 4.22).

The following claim establishes a relation between the weight of a tree, and the sum of coverings of its vertices.

Lemma 4.30 *For any ϑ -tree T of depth h , it holds that*

$$\sum_{e \in E(T)} w(e) = \sum_{v \in V(T)} \gamma(v) + n - 1.$$

Proof: For an edge $e \in E(T)$, let $\varphi(e)$ denote the number of vertices covered by e . Clearly, $w(e) = \varphi(e) + 1$. It is easy to verify by double counting that

$$\sum_{v \in V(T)} \gamma(v) = \sum_{e \in E(T)} \varphi(e).$$

Consequently,

$$\sum_{e \in E(T)} w(e) = \sum_{e \in E(T)} (\varphi(e) + 1) = \sum_{v \in V(T)} \gamma(v) + n - 1.$$

■

Denote the minimum covering of a ϑ -tree of depth *at most* h by $\tilde{\gamma}(n, h)$. Since the sequence $(\gamma(n, h))_{h=1}^{n-1}$ is monotone non-increasing (by Lemma 3.1), it holds that

$$\tilde{\gamma}(n, h) = \gamma(n, h). \quad (22)$$

The proof of the following lemma is closely related to the proof of Lemma 2.1 from [2], and is provided for completeness.

Lemma 4.31 *For a sufficiently large integer n , and a positive integer h , $h \leq \frac{1}{10} \cdot \log n$, it holds that*

$$W(n, h) = \Omega(h \cdot n^{1+1/h}).$$

Proof: Let T be a ϑ -tree of minimum weight $W(n, h)$, that has depth at most h , $h \leq \frac{1}{10} \cdot \log n$, and covering γ . Denote the root vertex of T by rt . Let $g(n, h) = \frac{1}{20} \cdot h \cdot n^{1/h}$. By Corollary 4.29 and (22),

$$\gamma > \tilde{\gamma}(n, h) \geq g(n, h). \quad (23)$$

A vertex v is called *heavy* if $\gamma(v) \geq \frac{1}{6} \cdot g(n, h)$, and it is called *light* otherwise. Let \mathcal{H} be the set of heavy vertices in T . Next, we prove that $|\mathcal{H}| \geq n/2$. Since by Lemma 4.30,

$$\sum_{e \in E(T)} w(e) \geq \sum_{v \in V(T)} \gamma(v) \geq |\mathcal{H}| \cdot \frac{1}{6} \cdot g(n, h),$$

this would complete the proof of the lemma.

Suppose to the contrary that $|\mathcal{H}| < n/2$. We need the following definition. A set $B = \{i, i+1, \dots, j\}$ of consecutive vertices is said to be a *block* if $B \subseteq \mathcal{H}$. If $(i-1) \notin B$ and $(j+1) \notin B$ then B is called a *maximal block*. Decompose \mathcal{H} into a set of (disjoint) maximal blocks $\mathcal{B} = \{B_1, B_2, \dots, B_k\}$. By contracting the induced subgraph of each B_i into a single node w_i , for $1 \leq i \leq k$, we obtain a new graph $G' = (V', E')$. Observe that G' is not necessarily a tree and may contain multiple edges connecting the same pair of vertices. For each pair of vertices u and v in V' , omit all edges except for the lightest one that connect u and v in E' . If $rt \in w_i$, for some index $i \in [1, k]$, then designate w_i as the new root vertex, denoted rt' . Let $T' = (V', E'')$ be the BFS tree of G' rooted at rt' . Clearly, $n' = |V'| \geq |V \setminus \mathcal{H}| > n/2$, and $h(T') \leq h(T) = h$. Let $v_1, v_2, \dots, v_{n'}$ be the vertices of T' in an increasing order. Transform T' into a spanning tree \tilde{T} of $\vartheta_{n'}$ by mapping each v_i to the point i on the x -axis, for $i = 1, 2, \dots, n'$.

Claim 4.32 $\gamma(\tilde{T}) \leq \frac{1}{3} \cdot g(n, h) + 1$.

Proof: First, observe that all light vertices in T remain light in \tilde{T} . For a contracted heavy vertex w_i , we argue that $\gamma(w_i) \leq \gamma(w_i^-) + \gamma(w_i^+) + 1$, where w_i^- (respectively, w_i^+) is the vertex to the immediate left (resp., right) of w_i . To see this, note that for each edge $e \neq (w_i^+, w_i^-)$ that covers w_i , e covers either w_i^- or w_i^+ . Since both vertices w_i^+ and w_i^- are light, it follows that $\gamma(w_i) \leq 2 \cdot (\frac{1}{6} \cdot g(n, h)) + 1$, and we are done. ■

Observe that for $n \geq 4$, we have that $\frac{1}{10} \cdot \log n \leq \frac{1}{5} \cdot \log(n/2)$. Since $h \leq \frac{1}{10} \cdot \log n$, and $n' > n/2$, it follows that $h \leq \frac{1}{5} \cdot \log n'$. Hence by Corollary 4.29 and (23),

$$\begin{aligned} \gamma(\tilde{T}) &\geq \tilde{\gamma}(n', h) \geq g(n', h) = \frac{1}{20} \cdot h \cdot n'^{\frac{1}{h}} \\ &> \frac{1}{20} \cdot h \cdot (n/2)^{\frac{1}{h}} = \left(\frac{1}{2}\right)^{\frac{1}{h}} \cdot g(n, h) \geq \frac{1}{2} \cdot g(n, h). \end{aligned}$$

However, for sufficiently large n ,

$$\frac{1}{2} \cdot g(n, h) > \frac{1}{3} \cdot g(n, h) + 1,$$

contradicting Claim 4.32. This completes the proof of Lemma 4.31. \blacksquare

Lemma 4.33 *For a sufficiently large integer n , and a positive integer h , $\frac{1}{10} \cdot \log n \leq h < \log n$, it holds that $W(n, h) = \Omega(h \cdot n^{1+1/h})$.*

Proof: By Theorem 4.22, the minimum weight $W(n, \log n)$ of a ϑ_n -tree that has depth h , $h \leq \log n$, is at least $\Omega(\Psi \cdot n)$, for some Ψ satisfying $\log n = \Omega(\Psi \cdot n^{1/\Psi})$. It follows that $W(n, \log n) = \Omega(\log n \cdot n)$. Since by Lemma 3.1, the sequence $(W(n, h))|_{h=1}^{n-1}$ is monotone non increasing, it holds that for $h \in [\frac{1}{10} \cdot \log n, \log n]$,

$$W(n, h) = \Omega(\log n \cdot n) = \Omega(h \cdot n^{1+1/h}).$$

\blacksquare

Lemmas 4.30 and 4.33 imply the following lower bound for $h < \log n$.

Corollary 4.34 *For a sufficiently large integer n , and a positive integer h , $h < \log n$, it holds that $W(n, h) = \Omega(h \cdot n^{1+1/h})$.*

Clearly, the lightness of a graph G is at least as large as that of any BFS tree of G , implying the following result.

Theorem 4.35 *For a sufficiently large integer n and a positive integer h , $h < \log n$, any spanning subgraph of ϑ_n with hop-radius at most h has weight at least $\Omega(h \cdot n^{1+1/h})$, and thus lightness at least $\Psi = \Omega(h \cdot n^{1/h})$.*

Corollary 4.23 and Theorem 4.35 provide together tight lower bounds for the tradeoff between the hop-radius and lightness for the entire range of parameters.

5 Euclidean Spanners

The following theorem settles the open problem of [6, 2]. It is a direct corollary of Theorem 4.22 and Corollary 4.34.

Theorem 5.1 *For a sufficiently large integer n , any spanning subgraph of ϑ that has hop-diameter at most $O(\log n)$ has lightness at least $\Omega(\log n)$, and vice versa.*

Proof: First, we show that any ϑ -tree that has depth at most $O(\log n)$ has lightness at least $\Omega(\log n)$. If the depth h is at least $\log n$, Theorem 4.22 implies that the lightness is at least $\Omega(\Psi)$, for some Ψ satisfying $h = \Omega(\Psi \cdot n^{1/\Psi})$. Observe that for $h = O(\log n)$, any Ψ that satisfies $h = \Omega(\Psi \cdot n^{1/\Psi})$ is at least $\Omega(\log n)$, as required. By the monotonicity (Lemma 3.1), the lower bound of $\Omega(\log n)$ applies for all smaller values of h .

Consider a spanning subgraph G of ϑ that has hop-diameter $H = O(\log n)$. Consider a BFS tree T rooted at some vertex rt . Obviously, $h(T, rt) \leq H = O(\log n)$, and thus $w(T) = \Omega(\log n) \cdot w(MST(\vartheta)) = \Omega(n \log n)$. Since $w(G) \geq w(T)$, it follows that the lightness of G is $\Omega(\log n)$ as well.

Next, we argue that any ϑ -tree that has lightness at most $O(\log n)$ has depth at least $\Omega(\log n)$. Indeed, by Corollary 4.34, any ϑ -tree of depth $h = o(\log n)$ has lightness at least $\Omega(h \cdot n^{1/h}) = \omega(\log n)$. Moreover, if a spanning subgraph G of ϑ has lightness at most $O(\log n)$, then its BFS tree T satisfies $\Psi(T) = O(\log n)$ as well. Therefore, $h(T, rt) = \Omega(\log n)$, and thus $H(G) \geq h(T, rt) = \Omega(\log n)$. ■

Theorem 5.1 implies that no construction that provides Euclidean spanners with hop-diameter $O(\log n)$ and lightness $o(\log n)$, or vice versa, is possible. This settles the open problem of [6, 2, 42, 45].

6 Upper Bounds

In this section we devise an upper bound for the tradeoff between various parameters of binary LLTs. This upper bound is tight up to constant factors in the entire range of parameters.

Consider a general n -point metric space M . Let T^* be an MST for M , and D be an in-order traversal of T^* , starting at an arbitrary vertex u . For every vertex x , remove from D all occurrences of x except for the first one. It is well-known (see, e.g., [26], ch. 36) that this way we obtain a Hamiltonian path $L = L(T)$ of M of total weight

$$w(L) = \sum_{e \in L} w(e) \leq 2 \cdot w(MST(M)).$$

Let u_1, u_2, \dots, u_n be the order in which the points of M appear in L , i.e., $(u_1, u_2, \dots, u_n) = L$. Consider an edge $e' = (u_i, u_j)$ connecting two arbitrary points in M , and an edge $e = (u_q, u_{q+1}) \in E(L)$, $q \in [n-1]$. The edge e' is said to *load* e (with respect to L) if $i \leq q < q+1 \leq j$. When L is clear from the context, we write that e' loads e .

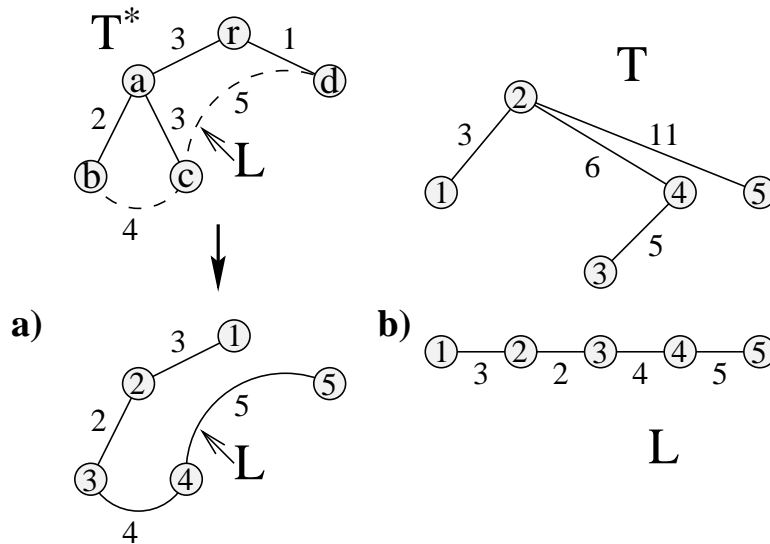


Figure 12: a) The construction of the Hamiltonian path L . b) The edges $(2, 4)$, $(2, 5)$, and $(3, 4)$ load the edge $(3, 4)$. Hence the load of T on the edge $(3, 4)$ is 3.

For a spanning tree T of M , the number of edges $e' \in E(T)$ that load an edge e of $E(L)$ is called the *load of e by T* and it is denoted $\xi(e) = \xi_T(e)$. The *load of the tree T* , $\xi(T)$, is the maximum load of an

edge $e \in E(L)$ by T , i.e.,

$$\xi(T) = \max\{\xi_T(e) \mid e \in E(L)\}.$$

By the triangle inequality,

$$w(T) \leq \sum_{e \in L(T)} \xi_T(e) \cdot w(e) \leq \xi(T) \cdot w(L),$$

and so,

$$\xi(T) \geq \frac{w(T)}{w(L)} \geq \frac{1}{2} \cdot \Psi(T). \quad (24)$$

(See Figure 12 for an illustration.)

In the sequel we provide an upper bound for the load $\xi(T)$ of a tree T , which yields the same upper bound for the lightness $\Psi(T)$ of T , up to a factor of 2. The next lemma shows that without loss of generality one can assume that u_i is located in the point i on the x -axis.

Lemma 6.1 *Suppose there exists a ϑ -tree T with load $\xi(T)$ and depth h . Then there exists a spanning tree T' for M with the same load (with respect to $L(T)$) and depth.*

Proof: The edge set E' of the tree T' is defined by $E' = \{(u_i, u_j) \mid (i, j) \in T\}$. It is easy to verify that its depth and load are equal to those of T . ■

Consequently, the problem of providing upper bounds for general metric spaces reduces to the problem of providing upper bounds for ϑ .

6.1 Upper Bounds for High Trees

In this section we devise a construction of ϑ_n -trees with depth $h \geq \log n$. This construction exhibits tight up to constant factors tradeoff between the depth and load, when the tree depth is at least logarithmic in the number of vertices. In addition, the constructed trees are *binary*, and so their maximum degree is *optimal*.

We start with defining a certain composition of binary trees. Let n' and n'' be two positive integers, $n = n' + n''$. Let ϑ' , ϑ'' , and ϑ be the n' -, n'' -, and n -point metric spaces $\vartheta_{n'}$, $\vartheta_{n''}$, and ϑ_n , respectively. Also, let $\{u'_1, u'_2, \dots, u'_{n'}\}$, $\{u''_1, u''_2, \dots, u''_{n''}\}$, and $\{\tilde{u}_1, \tilde{u}_2, \dots, \tilde{u}_n\}$ denote the set of points of ϑ' , ϑ'' , and ϑ , respectively. Consider spanning trees T' and T'' of ϑ' and ϑ'' , respectively. Let $u' = u'_i$ be a vertex of T' . Consider a tree \tilde{T} that spans the vertex set $\{\tilde{u}_1, \tilde{u}_2, \dots, \tilde{u}_{n'+n''}\}$ formed out of the trees T' and T'' in the following way. The root rt'' of T'' is added as a right child of u' in \tilde{T} . The vertices $u'_1, u'_2, \dots, u'_i = u'$ of T' retain their indices, and are translated into vertices $\tilde{u}_1, \tilde{u}_2, \dots, \tilde{u}_i$ in \tilde{T} . The vertices $u''_1, u''_2, \dots, u''_{n''}$ of T'' get the index i of u'_i , added to their indices, and are translated into vertices $\tilde{u}_{i+1}, \tilde{u}_{i+2}, \dots, \tilde{u}_{i+n''}$ in \tilde{T} . Finally, the vertices $u'_{i+1}, u'_{i+2}, \dots, u'_{n'}$ of T' get the number n'' of vertices of T'' added to their indices, and are translated into vertices $\tilde{u}_{i+1+n''}, \tilde{u}_{i+2+n''}, \dots, \tilde{u}_{n'+n''}$. We say that the tree \tilde{T} is composed by adding T'' as a right subtree to the vertex u' in T' . (See Figure 13 for an illustration.)

Adding a left subtree T'' to a vertex $u' = u'_i$ in T' is defined analogously. Specifically, the vertices $u'_1, u'_2, \dots, u'_{i-1}$ of T' retain their indices and become $\tilde{u}_1, \tilde{u}_2, \dots, \tilde{u}_{i-1}$, respectively. The vertices $u''_1, u''_2, \dots, u''_{n''}$ of T'' are translated into $\tilde{u}_{(i-1)+1} = \tilde{u}_i, \tilde{u}_{i+1}, \dots, \tilde{u}_{i+n''-1}$, respectively. Finally, the vertices $u'_i, u'_{i+1}, \dots, u'_{n'}$ of T' are translated into $\tilde{u}_{i+n''}, \tilde{u}_{i+n''+1}, \dots, \tilde{u}_{n'+n''}$, respectively.

Consider a family of binary ϑ -trees $T(\xi, h)$ with $n = N(\xi, h)$ vertices, load ξ and depth h , $h \geq \xi - 1$, $\xi \geq 1$. These trees are all rooted at the point 1. For $\xi = 1$, and $h = 0$, the tree $T(1, 0)$ is a singleton vertex, and so $N(1, 0) = 1$. For convenience, we define the load of $T(1, 0)$ to be 1. For $\xi = 1$ and $h \geq 1$, the tree $T(1, h)$ is the path $P_{h+1} = (u_1, u_2, \dots, u_{h+1})$. The depth of $T(1, h)$ is h , its load is 1, and its size $N(1, h)$ is $h + 1$.

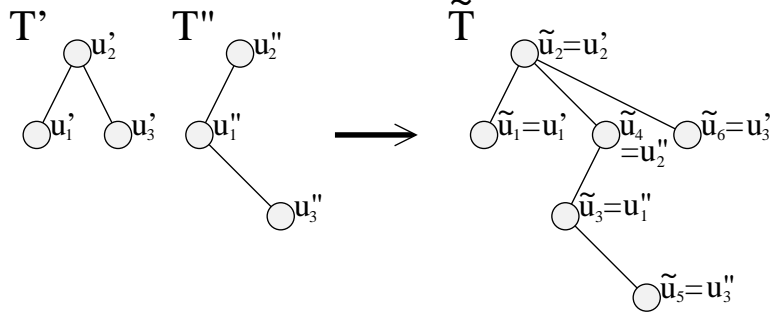


Figure 13: The trees T' and T'' depicted on the left are spanning trees of $\vartheta' = \vartheta'' = \vartheta_3$. The tree \tilde{T} depicted on the right is a spanning tree of $\vartheta = \vartheta_6$. It is composed by adding T'' as a right subtree to the vertex u'_2 in T' .

For $\xi \geq 2$ and $h \geq \xi - 1$, the tree $T(\xi, h)$ is constructed as follows. Let $T' = P_{h+1}$ be the path $(u_1, u_2, \dots, u_{h+1})$, and for each index i , $i \in [h]$, define for technical convenience $u'_i = u_i$. For each index i , $i \in [h]$, let T''_i be the tree $T(\xi''_i, h''_i)$, with $\xi''_i = \min\{\xi - 1, h - i + 1\}$, $h''_i = h - i$. (Observe that for each $i \in [h]$, $h''_i \geq \xi''_i - 1$, and thus the tree T''_i is well-defined.) For every $i \in [h]$, we add the tree T''_i as a right subtree of u'_i in T' . For $i \in [h+1]$, let $\tilde{u}^{(i)}$ be the image vertex of the vertex u'_i under this transformation. In other words, for each $i \in [h+1]$, the vertex u'_i of T' is translated into the vertex $\tilde{u}^{(i)}$ of the resulting tree $T(\psi, h)$. (Note that $\tilde{u}^{(1)} = u'_1$.) Let \tilde{P}_{h+1} denote the path $(u'_1 = \tilde{u}^{(1)}, \tilde{u}^{(2)}, \dots, \tilde{u}^{(h+1)})$ in $T(\psi, h)$. (See Figure 14 for an illustration.)

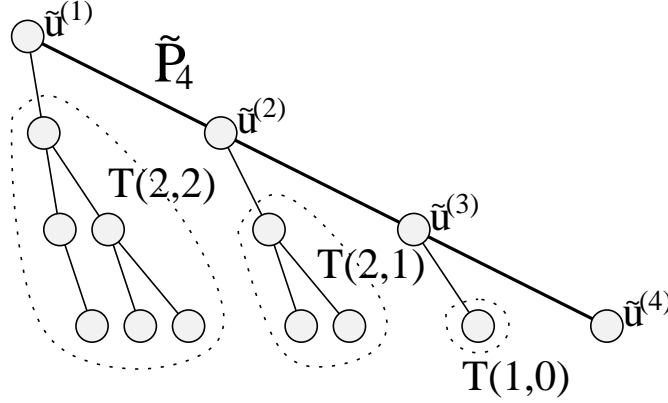


Figure 14: The tree $T(3, 3)$. The path \tilde{P}_4 is depicted by the thick line. The tree $T(2, 2)$ is added as a right subtree of $\tilde{u}^{(1)}$, and thus, its vertices are drawn to the right of $\tilde{u}^{(1)}$ but to the left of $\tilde{u}^{(2)}$.

Lemma 6.2 *The depth of the resulting tree $T(\xi, h)$ with respect to the vertex u'_1 is h , and its load is ξ .*

Proof: The proof is by induction on ξ , for all values of $h \geq \xi - 1$.

Base ($\xi = 1$): The tree $T(1, h)$ has load 1 and depth h , as required.

Induction Step: We assume the correctness of the statement for all smaller values of ξ , and prove it for ξ . First, the vertex $\tilde{u}^{(h+1)}$ is connected to the root $\tilde{u}^{(1)}$ via the path $\tilde{P}_{h+1} = (\tilde{u}^{(1)}, \tilde{u}^{(2)}, \dots, \tilde{u}^{(h+1)})$, and thus the depth of $T(\xi, h)$ is at least the length of this path, that is, h . Consider a vertex $w \in V(T''_i)$, for some $i \in [h]$. The path P_w connecting u'_1 and w in $T(\xi, h)$ starts with a subpath $(\tilde{u}^{(1)}, \tilde{u}^{(2)}, \dots, \tilde{u}^{(i)}, rt(T''_i))$,

and continues with the unique path P_w'' connecting between the root of T_i'' and the vertex w in the tree T_i'' . The length of the latter path is no greater than the depth h_i'' of T_i'' , which, by the induction hypothesis, is equal to $h - i$. Hence $|P_w| = i + |P_w''| \leq h$. Hence $h(T(\xi, h)) = h$.

To analyze the load of the tree $T(\xi, h)$, consider some edge $e = (u_q, u_{q+1})$ of the path $P_n = (u_1, u_2, \dots, u_n)$, where n is the number of vertices in $T(\xi, h)$. The path \bar{P}_{h+1} contributes at most one unit to the load of e . In addition, there exists at most one index $i, i \in [h]$, such that the tree T_i'' loads the edge e . By the induction hypothesis, $\xi(T_i'') = \xi_i'' = \min\{\xi - 1, h - i + 1\}$. Consequently, the load of e by $T(\xi, h)$ is

$$\begin{aligned} \xi_{T(\xi, h)}(e) &\leq \max\{\min\{\xi - 1, h - i + 1\} + 1 : i \in [h]\} \\ &= \min\{\xi - 1, h\} + 1 = \xi. \end{aligned}$$

Hence $\xi(T(\xi, h)) \leq \xi$. Also, consider an edge $e' = (u_{q'}, u_{q'+1})$, such that $\xi_{T_1''}(e') = \xi(T_1'') = \xi_1''$. (This is the most loaded edge by T_1'' .) It follows that

$$\xi(T(\xi, h)) \geq \xi_{T(\xi, h)}(e') \geq \xi_{T_1''}(e') + 1 = \xi_1'' + 1 = \min\{\xi - 1, h\} + 1 = \xi.$$

Thus $\xi(T(\xi, h)) = \xi$, completing the proof. \blacksquare

Finally, we analyze the number of vertices $N(\xi, h)$ in the tree $T(\xi, h)$. By construction,

$$N(\xi, h) = h + 1 + \sum_{i=1}^h N(\min\{\xi - 1, h - i + 1\}, h - i).$$

Lemma 6.3 For $h \geq \xi - 1$, $N(\xi, h) > \binom{h}{\xi}$.

Proof: The proof is by induction on ξ .

Base: For $\xi = 1$, $N(1, h) = h + 1 > h = \binom{h}{1}$, as required.

Step: For $\xi \geq 2$, and $i \leq h - \xi + 1$, $\min\{\xi - 1, h - i + 1\} = \xi - 1$. Hence

$$\begin{aligned} N(\xi, h) &= h + 1 + \sum_{i=1}^h N(\min\{\xi - 1, h - i + 1\}, h - i) \\ &> \sum_{i=1}^{h-\xi+1} N(\min\{\xi - 1, h - i + 1\}, h - i) = \sum_{i=1}^{h-\xi+1} N(\xi - 1, h - i). \end{aligned}$$

Observe that for each index $i, i \in [h - \xi + 1]$, $h - i \geq \xi - 1$. Hence, by the induction hypothesis and Fact A.1,

$$\sum_{i=1}^{h-\xi+1} N(\xi - 1, h - i) > \sum_{i=1}^{h-\xi+1} \binom{h-i}{\xi-1} = \binom{h}{\xi}.$$

\blacksquare

The following theorem summarizes the properties of the trees $T(\xi, h)$.

Theorem 6.4 For a sufficiently large n , and $h, h \geq 2\lceil \log n \rceil$, there exists a binary ϑ_n -tree that has depth at most h and load at most $\xi + 1$, where ξ satisfies $(h = O(\xi \cdot n^{1/\xi}))$ and $\xi = O(\log n)$.

Proof: By Lemma 6.3, for any pair of positive integers h and ξ such that $h \geq \xi - 1$, it holds that $N(\xi, h) > \binom{h}{\xi}$. Observe that $N(\xi, h)$ is monotone increasing with both ξ and h in the entire range $h \geq \xi - 1 \geq 0$,

and $N(1, h) = h + 1 \leq n$. Also, for h in this range, it holds that $N(\lceil \log n \rceil, h) > \binom{h}{\lceil \log n \rceil} \geq n$. It follows that for any $h \geq 2\lceil \log n \rceil$, there exists a positive integer ξ , $\xi \leq h$, such that $N(\xi, h) \leq n < N(\xi + 1, h)$.

Consider the binary tree T' obtained from the tree $T(\xi + 1, h)$ by removing $N(\xi + 1, h) - n$ leaves from it, one after another. (Each removed vertex is a leaf in the *current* tree T' . The tree T' is initialized as $T(\xi + 1, h)$, and is updated after each removal.) Clearly, the resulting tree T' has depth at most h , load at most $\xi + 1$, and its size is equal to n . Observe that

$$n \geq N(\xi, h) > \binom{h}{\xi} \geq \left(\frac{h}{\xi}\right)^\xi,$$

implying that

$$h < \xi \cdot n^{1/\xi} = O(\xi \cdot n^{1/\xi}).$$

Observe that T' does not span ϑ_n . Let $z_1 < z_2 < \dots < z_n$ be the sequence of vertices of T' in an increasing order. To transform T' into a spanning tree of ϑ_n , for each index i , $1 \leq i \leq n$, relocate z_i to the point i . The theorem follows. \blacksquare

The next corollary employs Theorem 6.4 to deduce an analogous result for general metric spaces.

Corollary 6.5 *For sufficiently large n , and h , $h \geq 2\lceil \log n \rceil$, there exists a binary spanning tree of M that has depth at most h and lightness at most $2 \cdot \Psi + 2$, where Ψ satisfies ($h = O(\Psi \cdot n^{1/\Psi})$ and $\Psi = O(\log n)$). Moreover, this binary tree can be constructed in time $O(n^2)$.*

Proof: By Theorem 6.4 and Lemma 6.1, for a sufficiently large n , and h , $h \geq 2\lceil \log n \rceil$, there exists a binary spanning tree T of M that has depth at most h and load at most $\Psi + 1$, where Ψ satisfies ($h = O(\Psi \cdot n^{1/\Psi})$ and $\Psi = O(\log n)$). By (24), the lightness $\Psi(T)$ of T is at most $2 \cdot \Psi + 2$. Note that since the complete graph $G(M)$ induced by the metric M contains at most $O(n^2)$ edges, its MST can be computed within $O(n^2)$ time (cf. [26], chapter 23). The in-order traversal, and the construction of the tree $T(\xi, h)$ can be performed in $O(n)$ time in the straight-forward way. Hence the overall running time of the algorithm for computing an LLT with the specified properties is $O(n^2)$. \blacksquare

Remark: If M is an Euclidean 2-dimensional metric space, the running time can be further improved to $O(n \cdot \log n)$. This is because the running time of this construction is dominated by the running time of the subroutine for constructing MST, and an MST of an Euclidean 2-dimensional metric space can be constructed in $O(n \cdot \log n)$ time [29]. By the same considerations, for Euclidean 3-dimensional spaces our algorithm can be implemented in a randomized time of $O(n \cdot \log^{4/3} n)$, and more generally, for dimension $d = 3, 4, \dots$ it can be implemented in deterministic time $O(n^{2 - \frac{2}{\lceil d/2 \rceil + 1} + \epsilon})$, for an arbitrarily small $\epsilon > 0$ (cf. [29], page 5). Even better time bounds can be shown if one uses a $(1 + \epsilon)$ -approximation MST instead of the exact MST for Euclidean metric spaces (cf. [29], page 6). Specifically, this technique enables us to obtain running time of $O(n \cdot \log n)$ for any constant dimension d .

Finally, we present a simple construction for the range $\log n \leq h < 2\lceil \log n \rceil$. Consider a full¹ balanced binary spanning tree τ_n of $\{1, 2, \dots, n\}$. The root of τ_n is $\lceil n/2 \rceil$. Its left (respectively, right) subtree is the full balanced binary tree constructed recursively from the vertex set $\{1, \dots, \lceil n/2 \rceil - 1\}$ (resp., $\{\lceil n/2 \rceil + 1, \dots, n\}$). (See Figure 15 for an illustration.)

Lemma 6.6 *Both the depth and the load of τ_n are no greater than $\log n$.*

¹Strictly speaking, this tree is full when $n = 2^k - 1$, for integer $k \geq 1$. However, for simplicity of presentation, we call it “full” for other values of n as well.

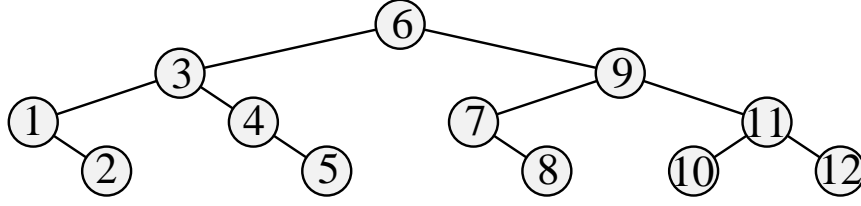


Figure 15: The full balanced binary tree τ_n for $n = 12$ and $h = 3$.

Proof: The proof is by induction on n . The base is trivial.

Induction Step: We assume the correctness of the claim for all smaller values of n , and prove it for n . By the induction hypothesis, the depth of both the left and the right subtrees of the root $\lceil n/2 \rceil$ is at most $\log(n/2)$, implying that the depth $h(\tau_n)$ of τ_n is at most $\log(n/2) + 1 = \log n$, as required.

To analyze the load $\xi(\tau_n)$ of the tree τ_n , consider some edge $e = (u_q, u_{q+1})$ of the path $P_n = (u_1, u_2, \dots, u_n)$. The two edges connecting rt to its children contribute at most one unit to the load of e . In addition, at most one among the subtrees of the root $\lceil n/2 \rceil$ loads the edge e . By induction, both of these subtrees have load no greater than $\log(n/2)$. Consequently, the load of e is at most $\log(n/2) + 1 = \log n$. ■

Theorem 6.7 *For a sufficiently large n , and h , $\log n \leq h < 2\lceil \log n \rceil$, there exists a binary ϑ_n -tree that has depth at most h and load at most ξ , where ξ satisfies ($h = O(\xi \cdot n^{1/\xi})$ and $\xi = O(\log n)$).*

Proof: By Lemma 6.6, for a sufficiently large n , and h , $\log n \leq h < 2\lceil \log n \rceil$, the binary tree τ_n has depth at most h and load at most $\xi = \log n$, where $h = O(\xi \cdot n^{1/\xi})$. ■

The next corollary employs Theorem 6.7 to deduce an analogous result for general metrics.

Corollary 6.8 *For sufficiently large n and h , $\log n \leq h < 2\lceil \log n \rceil$, there exists a binary spanning tree of M that has depth at most h and lightness at most $2 \cdot \Psi$, where Ψ satisfies ($h = O(\Psi \cdot n^{1/\Psi})$ and $\Psi = O(\log n)$).*

Proof: By Theorem 6.7 and Lemma 6.1, for a sufficiently large n , and h , $\log n \leq h < 2\lceil \log n \rceil$, there exists a binary spanning tree of M that has depth at most h and load at most Ψ , where Ψ satisfies ($h = O(\Psi \cdot n^{1/\Psi})$ and $\Psi = O(\log n)$). By (24), the lightness is at most twice the load, and we are done. ■

Remark: By the same considerations as those of Corollary 6.5, this construction can be implemented within time $O(n^2)$ in general metric spaces, and in time $O(n \cdot \log n)$ in Euclidean low-dimensional ones.

6.2 Upper Bounds for Low Trees

In this section we devise a construction of ϑ_n -trees with depth in the range of $h < \log n$. Like the construction in Section 6.1, this construction provides a tight *up to constant factors* upper bound for the tradeoff between the depth and lightness. In addition, the maximum degree of the constructed trees is optimal as well. (It is $\lceil n^{1/h} \rceil + 1$).

Consider a family of d -ary ϑ -trees $\tilde{T}(d, h)$ with $\tilde{n} = \tilde{N}(d, h) = \sum_{i=0}^h d^i$ vertices, load at most $d \cdot h$ and depth h , $d \geq 1$. (For $d \geq 2$ and any h , $\tilde{n} = \frac{d^{h+1}-1}{d-1}$. It is easy to verify that for $d \geq 2$ and $h \geq 1$,

$h < \log_d \tilde{n} \leq \log \tilde{n}$.) The family $\tilde{T}(d, h)$ is constructed recursively as follows. For $h = 0$, the tree $T_0 = \tilde{T}(d, 0)$ is a singleton vertex, and so $\tilde{N}(d, 0) = 1$. For $h \geq 1$, the tree $\tilde{T}(d, h)$ is constructed as follows. For each $i \in [d]$, let T_i be a copy of $\tilde{T}(d, h - 1)$, and let $T_0 = \tilde{T}(d, 0)$. [[S: In the previous section we denoted the full balanced binary tree by T_n , so we better find another name for it; I suggest to call it F_n instead.]] For every $i \in \lceil [d/2] \rceil$, we add the tree T_i as a *left* subtree of the root vertex rt of T_0 , and for each $i \in \lceil [d/2] + 1, d \rceil$, we add the tree T_i as a *right* subtree of rt .

Lemma 6.9 *The depth of the resulting tree $\tilde{T}(d, h)$ is h , its load is at most $d \cdot h$ and its size is $\sum_{i=0}^h d^i$. Moreover, the tree $\tilde{T}(d, h)$ is d -ary.*

Proof: The proof is by induction on h . The base $h = 0$ is trivial.

Induction Step: We assume the correctness of the claim for all smaller values of h , and prove it for h . By construction, the root rt has d children, each being the root of a tree $\tilde{T}(d, h - 1)$. By the induction hypothesis, $\tilde{T}(d, h - 1)$ is a d -ary tree of depth $h - 1$ and size $\sum_{i=0}^{h-1} d^i$. Hence $\tilde{T}(d, h)$ is a d -ary tree of depth h . To estimate the size $\tilde{N}(d, h)$ of $\tilde{T}(d, h)$, note that

$$\tilde{N}(d, h) = 1 + d \cdot \tilde{N}(d, h - 1) = 1 + d \cdot \left(\sum_{i=0}^{h-1} d^i \right) = \sum_{i=0}^h d^i.$$

To analyze the load of the tree $\tilde{T}(d, h)$, consider some edge $e = (u_q, u_{q+1})$ of the path $P_{\tilde{n}}$, where $\tilde{n} = \tilde{N}(d, h)$. The d edges connecting rt to its children contribute altogether at most d units to the load of e . In addition, there exists at most one index $i, i \in [h]$, such that the tree T_i loads the edge e . By the induction hypothesis, the load of T_i is no greater than $d(h - 1)$. Consequently, the load of e is at most $d(h - 1) + d = d \cdot h$. ■

Theorem 6.10 *For any positive integers n and $h, h < \log n$, there exists a ϑ_n -tree that has depth at most h , load at most $h \cdot \lceil n^{1/h} \rceil$ and maximum degree $\lceil n^{1/h} \rceil + 1$.*

Proof: The case $h = 0$ is trivial.

For $h > 0$, observe that $\tilde{N}(d, h)$ is monotone increasing with both d and h , and $\tilde{N}(1, h) = h + 1 \leq n$. Given a pair of positive integers n and $h, 1 \leq h < \log n$, let d be the positive integer that satisfies $\tilde{N}(d - 1, h) \leq n < \tilde{N}(d, h)$. (Note that $d \geq 2$.) We have that

$$n \geq \tilde{N}(d - 1, h) = \sum_{i=0}^h (d - 1)^i > (d - 1)^h,$$

implying that $d < n^{1/h} + 1$. Thus $d \leq \lceil n^{1/h} \rceil$.

Consider the tree T' obtained from $\tilde{T}(d, h)$ by removing $\tilde{N}(d, h) - n$ leaves from it, one after another. (Each removed vertex is a leaf in the *current* tree T' . The tree T' is initialized as $\tilde{T}(d, h)$, and is updated after each removal.) Clearly, the depth and the load of the resulting tree T' are no greater than those of the original tree $\tilde{T}(d, h)$. Hence by Lemma 6.9, the size of the resulting tree T' is n , its depth is at most h and its load is at most $h \cdot \lceil n^{1/h} \rceil$. Moreover, since $\tilde{T}(d, h)$ is a d -ary tree, so is T' , implying that its maximum degree is at most $d + 1 \leq \lceil n^{1/h} \rceil + 1$.

Observe that T' does not span ϑ_n . Let $z_1 < z_2 < \dots < z_n$ be the sequence of vertices of T' in an increasing order. To transform T' into a spanning tree of ϑ_n , for each index $i, 1 \leq i \leq n$, relocate z_i to the point i . The theorem follows. ■

The next corollary is an extension of Theorem 6.10 to general metric spaces.

Corollary 6.11 *For a sufficiently large n , and h , $h < \log n$, there exists a spanning tree of M that has depth at most h , lightness at most $O(h \cdot n^{1/h})$ and maximum degree at most $\lceil n^{1/h} \rceil + 1$.*

Proof: By Theorem 6.10 and Lemma 6.1, for a sufficiently large n , and h , $h < \log n$, there exists a spanning tree of M that has depth at most h , load at most $h \cdot \lceil n^{1/h} \rceil$ and maximum degree at most $\lceil n^{1/h} \rceil + 1$. By (24), the lightness is at most twice the load, and we are done. ■

Remark: By the same considerations as in Section 6.1, this construction can be implemented within time $O(n^2)$ in general metric spaces, and in time $O(n \cdot \log n)$ in Euclidean low-dimensional ones.

Corollaries 6.5, 6.11 and 6.8 imply the following theorem.

Theorem 6.12 *For any sufficiently large integer n and positive integer h , and n -point metric space M , there exists a spanning tree of M of depth at most h and lightness at most $O(\Psi)$, that satisfies the following relationship. If $h \geq \log n$ then ($h = O(\Psi \cdot n^{1/\Psi})$ and $\Psi = O(\log n)$). In the complementary range $h < \log n$, $\Psi = O(h \cdot n^{1/h})$.*

Moreover, this spanning tree is a binary one for $h \geq \log n$, and it has the optimal maximum degree $\lceil n^{1/h} \rceil + 1$, for $h < \log n$.

In view of Corollary 4.23 and Theorem 4.35, these upper bounds are tight up to constant factors.

Acknowledgements

The second-named author thanks Michael Segal and Hanan Shpungin for approaching him with a problem in the area of wireless networks that is related to the problem of constructing LLTs. Correspondence with them triggered this research. Also, we are grateful to Guy Kortsarz for his helpful and timely comments on a preliminary draft of this paper.

References

- [1] I. Abraham and D. Malkhi. Compact routing on euclidian metrics. In *Proc. of 23rd Annual Symp. on Principles of Distributed Computing*, 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, R. M. Karp, D. Peleg, and D. B. West. A graph-theoretic game and its application to the k -server problem. *SIAM J. Comput.*, 24(1):78–100, 1995.
- [4] C. J. Alpert, T. C. Hu, J. H. Huang, A. B. Kahng, and D. Karger. Prim-dijkstra tradeoffs for improved performance-driven routing tree design. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 14(7):890–896, 1995.
- [5] E. Althaus, S. Funke, S. Har-Peled, J. Könnemann, E. A. Ramos, and M. Skutella. Approximating k -hop minimum-spanning trees. *Oper. Res. Lett.*, 33(2):115–120, 2005.
- [6] 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.
- [7] 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.
- [8] S. Arya and M. H. M. Smid. Efficient construction of a bounded degree spanner with low weight. *Algorithmica*, 17(1):33–54, 1997.
- [9] B. Awerbuch and D. P. A. Baratz. Efficient broadcast and light-weight spanners. *Manuscript*, 1991.
- [10] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. In *Proc. of 9th PODC*, pages 177–187, 1990.

- [11] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proc. of 37th FOCS*, pages 184–193, 1996.
- [12] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proc. of 30th STOC*, pages 161–168, 1998.
- [13] K. Bharath-Kumar and J. M. Jaffe. Routing to multiple destinations in computer networks. *IEEE Trans. on Commun.*, COM-31:343–351, 1983.
- [14] P. Bose, J. Gudmundsson, and M. H. M. Smid. Constructing plane spanners of bounded degree and low weight. *Algorithmica*, 42(3-4):249–264, 2005.
- [15] P. B. Callahan and S. R. Kosaraju. A decomposition of multi-dimensional point-sets with applications to k -nearest-neighbors and n -body potential fields. In *Proc. of 24th STOC*, pages 546–556, 1992.
- [16] H. T.-H. Chan and A. Gupta. Small hop-diameter sparse spanners for doubling metrics. In *Proc. of 17th SODA*, pages 70–78, 2006.
- [17] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. *J. Algorithms*, 33(1):73–91, 1999.
- [18] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. A. Plotkin. Approximating a finite metric by a small number of tree metrics. In *Proc. of 39th FOCS*, pages 379–388, 1998.
- [19] M. Charikar, M. T. Hajiaghayi, H. J. Karloff, and S. Rao. l_2^2 spreading metrics for vertex ordering problems. In *Proc. of 17th SODA*, pages 1018–1027, 2006.
- [20] B. Chazelle and B. Rosenberg. The complexity of computing partial sums off-line. *Int. J. Comput. Geom. Appl.*, 1:33–45, 1991.
- [21] E. Cohen. Fast algorithms for constructing t -spanners and paths with stretch t . In *Proc. of 34th FOCS*, pages 648–658, 1993.
- [22] E. Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. In *Proc. of 26th STOC*, pages 16–26, 1994.
- [23] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong. Performance-driven global routing for cell based ics. In *Proc. of IEEE International Conference on Computer Design: VLSI in Computer & Processors (ICCD)*, pages 170–173, 1991.
- [24] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong. Provably good algorithms for performance-driven global routing. In *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2240–2243, 1992.
- [25] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong. Provably good performance-driven global routing. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 11(6):739–752, 1992.
- [26] T. H. Corman, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd edition*. McGraw-Hill Book Company, Boston, MA, 2001.
- [27] G. Das and G. Narasimhan. A fast algorithm for constructing sparse Euclidean spanners. In *Proc. of 10th SOCG*, pages 132–139, 1994.
- [28] M. Elkin, Y. Emek, D. Spielman, and S. Teng. Lower stretch spanning trees. In *Proc. of 37th STOC*, pages 494–503, 2005.
- [29] D. Eppstein. Spanning trees and spanners. *Technical report, Dept. of Information and Computer-Science, University of California, Irvine*, (96-16), 1996.
- [30] G. Even, J. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. In *Proc. of 36th FOCS*, pages 62–71, 1995.
- [31] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proc. of 35th STOC*, pages 448–455, 2003.
- [32] U. Feige and J. R. Lee. An improved approximation ratio for the minimum linear arrangement problem. *Inf. Process. Lett.*, 101(1):26–29, 2007.
- [33] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [34] L. Gouveia and T. L. Magnanti. Network flow models for designing diameter-constrained minimum-spanning and Steiner trees. *Networks*, 41(3):159–173, 2003.
- [35] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, 1994.
- [36] Y. Hassin and D. Peleg. Sparse communication networks and efficient routing in the plane. In *Proc. of 19th PODC*, pages 41–50, 2000.
- [37] J. M. Jaffe. Distributed multi-destination routing: the constraints of local information. *SIAM J. Comput.*, 14:875–888, 1985.

- [38] E. Kantor and D. Peleg. Approximate hierarchical facility location and applications to the shallow Steiner tree and range assignment problems. In *Proc. of 6th CIAC*, pages 211–222, 2006.
- [39] S. Khuller, B. Raghavachari, and N. E. Young. Approximating the minimum equivalent diagraph. In *Proc. of 5th SODA*, pages 177–186, 1994.
- [40] G. Kortsarz and D. Peleg. Approximating the weight of shallow Steiner trees. *Discrete Applied Mathematics*, 93(2-3):265–285, 1999.
- [41] S. Laue and D. Matijevic. Approximating k -hop minimum spanning trees in Euclidean metrics. In *Proc. of 19th CCCG*, pages 117–120, 2007.
- [42] H. P. Lenhof, J. S. Salowe, and D. E. Wrege. New methods to mix shortest-path and minimum spanning trees. manuscript, 1994.
- [43] Y. Mansour and D. Peleg. An approximation algorithm for min-cost network design. *DIMACS Series in Discr. Math and TCS*, 53:97–106, 2000.
- [44] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. H. III. Bicriteria network design problems. *J. Algorithms*, 28(1):142–171, 1998.
- [45] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.
- [46] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, Philadelphia, PA, 2000.
- [47] M. Pătraşcu and E. D. Demaine. Tight bounds for the partial-sums problem. In *Proc. of 15th SODA*, pages 20–29, 2004.
- [48] S. Rao and A. W. Richa. New approximation techniques for some ordering problems. In *Proc. of 9th SODA*, pages 211–218, 1998.
- [49] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning trees short or small. In *Proc. of 5th SODA*, pages 546–555, 1994.
- [50] A. C. Yao. Space-time tradeoff for answering range queries. In *Proc. of 14th STOC*, pages 128–136, 1982.

Appendix

A Properties of the Binomial Coefficients

In this section we present a number of useful properties of the binomial coefficients.

The following statement is well-known [35].

Fact A.1 (Pascal's 2nd identity) *For any non-negative integers h and i , such that $i \leq h$,*

$$\sum_{k=i}^h \binom{k}{i} = \binom{h+1}{i+1}.$$

The next lemma shows that the sequence of binomial coefficients $\mathcal{B} = \{\binom{n}{i} \mid i = 1, 2, \dots, n\}$ grows exponentially with i as long as $i \leq \lfloor \frac{n}{4} \rfloor$.

Lemma A.2 *For any non-negative integers n and k , such that $k \leq \lfloor \frac{n}{4} \rfloor$,*

$$\sum_{i=0}^k \binom{n}{i} < \frac{3}{2} \cdot \binom{n}{k}.$$

Proof: For any $0 \leq i \leq k-1$, it holds that

$$\binom{n}{i+1} = \frac{n-i}{i+1} \cdot \binom{n}{i} \geq \frac{n - \lfloor \frac{n}{4} \rfloor + 1}{\lfloor \frac{n}{4} \rfloor} \cdot \binom{n}{i} > 3 \cdot \binom{n}{i}.$$

Hence,

$$\sum_{i=0}^k \binom{n}{i} \leq \binom{n}{k} \cdot \sum_{i=0}^k \left(\frac{1}{3}\right)^i < \frac{3}{2} \cdot \binom{n}{k}.$$

■

Next, we prove a number of auxiliary technical statements.

Claim A.3 *For $h > \lfloor 2 \log n \rfloor$, $\binom{h+1}{\lfloor \log n \rfloor} > \frac{2}{3} \cdot n$.*

Proof: The statement clearly holds for $n = 1$. We henceforth assume that $n \geq 2$. For $h > \lfloor 2 \log n \rfloor$,

$$\binom{h+1}{\lfloor \log n \rfloor} \geq \binom{\lfloor 2 \log n \rfloor + 2}{\lfloor \log n \rfloor} \geq \binom{2 \lfloor \log n \rfloor + 2}{\lfloor \log n \rfloor}.$$

Observe that for any integers a and b , $a \geq b$, it holds that

$$\binom{a+1}{b} = \frac{a+1}{a+1-b} \cdot \binom{a}{b}$$

and

$$\binom{a}{b} \geq \left(\frac{a}{b}\right)^b.$$

It follows that

$$\begin{aligned}
\binom{2\lfloor \log n \rfloor + 2}{\lfloor \log n \rfloor} &= \frac{2\lfloor \log n \rfloor + 2}{\lfloor \log n \rfloor + 2} \cdot \frac{2\lfloor \log n \rfloor + 1}{\lfloor \log n \rfloor + 1} \cdot \binom{2\lfloor \log n \rfloor}{\lfloor \log n \rfloor} \\
&\geq 2 \cdot \frac{2\lfloor \log n \rfloor + 1}{\lfloor \log n \rfloor + 2} \cdot 2^{\lfloor \log n \rfloor} \\
&\geq 2 \cdot \frac{2\lfloor \log n \rfloor + 1}{\lfloor \log n \rfloor + 2} \cdot 2^{\log n - 1} = n \cdot \frac{2\lfloor \log n \rfloor + 1}{\lfloor \log n \rfloor + 2}.
\end{aligned}$$

It is easy to verify that for $n \geq 2$, $\frac{2\lfloor \log n \rfloor + 1}{\lfloor \log n \rfloor + 2} \geq 1$, implying that

$$\binom{h+1}{\lfloor \log n \rfloor} \geq \binom{2\lfloor \log n \rfloor + 2}{\lfloor \log n \rfloor} \geq n > \frac{2}{3} \cdot n.$$

□

Claim A.4

$$n \leq \binom{\lfloor 2 \log n \rfloor + 1}{\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \rfloor + 1}.$$

Proof: It is easy to verify the validity of the claim for $n \leq 15$. We henceforth assume that $n \geq 16$. Observe that for a positive number a , the function $f(x) = \left(\frac{a}{x}\right)^x$ is monotone increasing with x in the interval $\{x : 0 \leq x < a/e\}$. It is easy to show that for $n \geq 16$,

$$\left\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \right\rfloor + 1 < \frac{\lfloor 2 \log n \rfloor + 1}{e}. \tag{25}$$

It follows that

$$\left(\frac{\lfloor 2 \log n \rfloor + 1}{\left\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \right\rfloor + 1} \right)^{\left\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \right\rfloor + 1} \geq \left(\frac{\lfloor 2 \log n \rfloor + 1}{\frac{\lfloor 2 \log n \rfloor + 1}{4}} \right)^{\frac{\lfloor 2 \log n \rfloor + 1}{4}},$$

and consequently,

$$\begin{aligned}
\binom{\lfloor 2 \log n \rfloor + 1}{\left\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \right\rfloor + 1} &\geq \left(\frac{\lfloor 2 \log n \rfloor + 1}{\left\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \right\rfloor + 1} \right)^{\left\lfloor \frac{\lfloor 2 \log n \rfloor + 1}{4} \right\rfloor + 1} \\
&\geq \left(\frac{\lfloor 2 \log n \rfloor + 1}{\frac{\lfloor 2 \log n \rfloor + 1}{4}} \right)^{\frac{\lfloor 2 \log n \rfloor + 1}{4}} = 4^{\frac{\lfloor 2 \log n \rfloor + 1}{4}} \\
&\geq 4^{\frac{2 \log n}{4}} = n.
\end{aligned}$$

□

B A Lower Bound for Graphs

In this section we show that there are graphs for which any spanning tree has either huge hop-diameter or huge weight. Specifically, consider the graph $G = (V, E, w)$ formed as union of the $(n - 1)$ -vertex path $P_{n-1} = (v_1, v_2, \dots, v_{n-1})$ with the star $S = \{(z, v_i) | i \in [n - 1]\}$. All edges of the path have unit weight, and all edges of S have weight W , for some large integer W , $n \ll W$. (See Figure 16 for an illustration.) Note that G is a *metric graph*, that is, for every edge $(u, w) \in E$, $w(u, w) = \text{dist}_G(u, w)$. It is easy to verify that any spanning tree T of G that has weight at most $q \cdot W$, for some integer parameter q , $q \geq 1$, has hop-diameter $\Omega(n/q)$. Symmetrically, for an integer parameter D , $D \geq 1$, any spanning tree T of G that has hop-diameter at most D contains at least $\Omega(n/D)$ edges of weight W . Since the weight of the MST of G is only slightly greater than W , it follows that the lightness of T is at least $\Omega(n/D)$.

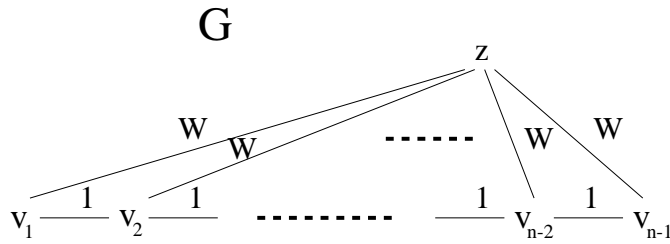


Figure 16: The graph $G = (V, E, w)$.

Consequently, for any general construction of LLTs for *graphs*, $\Psi \cdot H = \Omega(n)$. As we have shown, for *metric spaces* the situation is drastically better, and, in particular, one can have both Ψ and H no greater than $O(\log n)$.