Algorithmic and Hardness Results for the Hub Labeling Problem

Haris Angelidakis* Yury Makarychev* Vsevolod Oparin†

Abstract

There has been significant success in designing highly efficient algorithms for distance and shortest-path queries in recent years; many of the state-of-the-art algorithms use the hub labeling framework. In this paper, we study the approximability of the Hub Labeling problem. We prove a hardness of $\Omega(\log n)$ for Hub Labeling, matching known approximation guarantees. The hardness result applies to graphs that have multiple shortest paths between some pairs of vertices. No hardness of approximation results were known previously. Then, we focus on graphs that have a unique shortest path between each pair of vertices. This is a very natural family of graphs, and much research on the Hub Labeling problem has studied such graphs. We give an $O(\log D)$ approximation algorithm for graphs of shortest-path diameter D with unique shortest paths. In particular, we get an $O(\log \log n)$ approximation for graphs of polylogarithmic diameter, while previously known algorithms gave an $O(\log n)$ approximation. Finally, we present a polynomial-time approximation scheme (PTAS) and quasi-polynomial-time algorithms for Hub Labeling on trees; additionally, we analyze a simple combinatorial heuristic for Hub Labeling on trees, proposed by Peleg in 2000. We show that this heuristic gives an approximation factor of 2.

1 Introduction

There has been significant success in designing highly efficient algorithms for distance and shortest-path queries in recent years; many of the state-of-the-art algorithms use the hub labeling framework. In this paper, we present approximation algorithms as well as prove hardness results for the Hub Labeling problem. The Hub Labeling problem was introduced by Cohen et al.[1] in 2003 [15].

DEFINITION 1.1. (HUB LABELING) *Consider an undirected graph $G = (V, E)$ with edge lengths $l(e) > 0$. Suppose that we are given a set system $\{H_u\}_{u \in V}$ with one set $H_u \subset V$ for every vertex u. We say that $\{H_u\}_{u \in V}$ is a hub labeling if it satisfies the following covering property: for every pair of vertices (u, v) (u and v are not necessarily distinct), there is a vertex in $H_u \cap H_v$ (a common "hub" for u and v) that lies on a shortest path between u and v. We call vertices in sets H_u hubs: a vertex $v \in H_u$ is a hub for u.*

In the Hub Labeling problem (HL), our goal is to find a hub labeling with a small number of hubs; specifically, we want to minimize the ℓ_p-cost of a hub labeling.

DEFINITION 1.2. *The ℓ_p-cost of a hub labeling $\{H_u\}_{u \in V}$ equals $(\sum_{u \in V} |H_u|^p)^{1/p}$ for $p \in [1, \infty)$; the ℓ_∞-cost is $\max_{u \in V} |H_u|$. The hub labeling problem with the ℓ_p-cost, which we denote by HL_p, asks to find a hub labeling with the minimum possible ℓ_p-cost.*

Note: *When we talk about HL and do not specify the cost function explicitly, we mean HL_1; we sometimes refer to the ℓ_1-cost of $\{H_u\}_{u \in V}$ simply as the cost of $\{H_u\}_{u \in V}$.*

We are interested in the Hub Labeling problem because of its connection to the shortest-path problem. Nowadays hundreds of millions of people worldwide use web mapping services and GPS devices to get driving directions. That creates a huge demand for fast algorithms for computing shortest paths (algorithms that are even faster than the classic Dijkstra's algorithm). Hub labelings provide a highly efficient way for computing shortest paths (see also the paper of Bast et al. [10] for a review and discussion of various methods for computing shortest paths that are used in practice).

Let us briefly explain the connection between the Hub Labeling and shortest-path problems. Consider a graph $G = (V, E)$ with edge lengths $l(e)$. Let $d(u, v)$ be the shortest-path metric on G. Suppose that we have a hub labeling $\{H_u\}_u$. During the preprocessing step, we compute and store the distance $d(u, w)$ between every vertex u and each hub $w \in H_u$ of u. Observe that now we can very quickly answer a distance query: to find $d(u, v)$ we compute $\min_{w \in H_u \cap H_v}(d(u, w) + d(v, w))$. By the triangle inequality, $d(u, v) \le \min_{w \in H_u \cap H_v}(d(u, w) + d(v, w))$, and the covering property guarantees that there is a hub $w \in H_u \cap H_v$ on a shortest path between u and v; so $d(u, v) = \min_{w \in H_u \cap H_v}(d(u, w) + d(v, w))$. We can compute $\min_{w \in H_u \cap H_v}(d(u, w) + d(v, w))$ and answer the distance query in time $O(\max(|H_u|, |H_v|))$. We need to keep a lookup table of size $O(\sum_{u \in V} |H_u|)$

*Toyota Technological Institute at Chicago.

†Saint Petersburg Academic University of the Russian Academy of Sciences.

[1] See also prior papers on bit labeling schemes [13, 14, 22, 20, 27, 24, 18].

to store the distances between the vertices and their hubs. So, if, say, all hub sets H_u are of polylogarithmic size, the algorithm answers a distance query in polylogarithmic time and requires $n \operatorname{polylog} n$ space. The outlined approach can be used not only for computing distances but also shortest paths between vertices. It is clear from this discussion that it is important to have a hub labeling of small size, since both the running time and storage space depend on the number of hubs.

Recently, there has been a lot of research on algorithms for computing shortest paths using the hub labeling framework (see e.g. the following papers by Abraham et al. [6, 4, 3, 5, 1, 2]). It was noted that these algorithms perform really well in practice (see e.g. [4]). A systematic attempt to explain why this is the case led to the introduction of the notion of *highway dimension* [6]. Highway dimension is an interesting concept that managed to explain, at least partially, the success of the above methods: it was proved that graphs with small highway dimension have hub labelings with a small number of hubs; moreover, there is evidence that most real-life road networks have low highway dimension [11].

However, most papers on HL offer only algorithms with absolute guarantees on the cost of the hub labeling they find (e.g., they show that a graph with a given highway dimension has a hub labeling of a certain size and provide an algorithm that finds such a hub labeling); they do not relate the cost of the hub labeling to the cost of the optimal hub labeling. There are very few results on the approximability of the Hub Labeling problem. Only very recently, Babenko et al. [9] and White [26] proved respectively that HL_1 and HL_∞ are NP-hard. Cohen et al. [15] gave an $O(\log n)$-approximation algorithm for HL_1 by reducing the problem to a Set Cover instance and using the greedy algorithm for Set Cover to solve the obtained instance (the latter step is nontrivial since the reduction gives a Set Cover instance of exponential size); later, Babenko et al. [8] gave a combinatorial $O(\log n)$-approximation algorithm for HL_p, for any $p \in [1, \infty]$.

Our Results. In this paper, we obtain the following results. We prove an $\Omega(\log n)$ hardness for HL_1 and HL_∞ on graphs that have multiple shortest paths between some pairs of vertices (assuming that P $\neq$ NP). The result shows that the algorithms by Cohen et al. and Babenko et al. are asymptotically optimal. Since it is impossible to improve the approximation guarantee of $O(\log n)$ for arbitrary graphs, we focus on special families of graphs. We consider the family of graphs with *unique shortest paths* — graphs in which there is only one shortest path between every pair of vertices. This family of graphs appears in the majority of prior works on hub labeling (see e.g. [1, 9, 5]) and is very natural, in our opinion, since in real life all edge lengths are somewhat random, and, therefore, any two paths between two vertices u and v have different lengths. For such graphs, we design an approximation algorithm with approximation guarantee $O(\log D)$, where D is the shortest-path diameter of the graph (which equals the maximum hop length of a shortest path; see Section 2.1 for the definition); the algorithm works for every fixed $p \in [1, \infty)$ (the constant in the O-notation depends on p). In particular, this algorithm gives an $O(\log \log n)$ factor approximation for graphs of diameter $\operatorname{polylog} n$, while previously known algorithms give only an $O(\log n)$ approximation. Our algorithm crucially relies on the fact that the input graph has unique shortest paths; in fact, our lower bounds of $\Omega(\log n)$ on the approximation ratio apply to graphs of constant diameter (with non-unique shortest paths). We also extensively study HL on trees. Somewhat surprisingly, the problem is not at all trivial on trees. In particular, the standard LP relaxation for the problem is not integral. We obtain the following results for trees.

1. Design a polynomial-time approximation scheme (PTAS) for HL_p for every $p \in [1, \infty]$.
2. Design an exact quasi-polynomial time algorithm for HL_p for every $p \in [1, \infty]$, with the running time $n^{O(\log n)}$ for $p \in \{1, \infty\}$ and $n^{O(\log^2 n)}$ for $p \in (1, \infty)$.
3. Analyze a simple combinatorial heuristic for trees, proposed by Peleg in 2000, and prove that it gives a 2-approximation for HL_1 (we also show that this heuristic does not work well for HL_p when p is large).

Organization and overview of results. We first present an $O(\log D)$ approximation algorithm for graphs with unique shortest paths (see Sections 2 and 4). The algorithm solves a natural linear programming (LP) relaxation for the problem. Then, it uses the LP solution to find a system of "pre-hubs" $\{\widehat{H}_u\}_{u \in V}$ — pre-hubs do not necessarily satisfy the covering property (which hubs must satisfy), instead, they satisfy a "weak" covering property (see Definition 3.1 for details). The cost of the pre-hub labeling is at most 2 times the LP value. Finally, the algorithm converts the pre-hub labeling $\{\widehat{H}_u\}_{u \in V}$ to a hub labeling $\{H_u\}_{u \in V}$. To this end, it runs a randomized "correlated" rounding scheme (which always finds a feasible hub labeling). The expected cost of the obtained labeling is at most $O(\log D)$ times the cost of the pre-hub labeling. Thus, the algorithm finds an $O(\log D)$-approximate solution. We present the algorithm for HL_1 in Section 4 and

the algorithm for general HL_p in Appendix A (the algorithms for HL_1 and HL_p are almost identical, but the analysis for HL_1 is slightly simpler).

A key property of our rounding procedure that we would like to emphasize is that it might add a vertex v to a hub set H_u, even if the corresponding LP indicator variable x_{uv} (which, in an intended integral solution, is 1 if and only if $v \in H_u$) is set to zero in the fractional solution that we consider. This may seem surprising since often rounding schemes satisfy the following natural property: the rounding scheme adds an element to a set or makes an assignment only if the LP indicator variable for the corresponding event is strictly positive. In particular, known rounding schemes for Set Cover, as well as many other thresholding and randomized rounding schemes, satisfy this property (we call such rounding schemes "natural"). However, we show in Appendix B that any rounding scheme that satisfies this property cannot give a better than $\Theta(\log n)$ approximation for HL_1.

Then, in Section 5, we present our algorithms for trees. We consider a special class of hub labelings — *hierarchical hub labelings* — introduced by Abraham et al. [5] (we define and briefly discuss hierarchical hub labelings in Section 2.3). We start with proving that there is an optimal solution for a tree, which is a hierarchical hub labeling. We observe that there is a simple dynamic program (DP) of exponential size for computing the optimal hierarchical hub labeling on a tree: roughly speaking, we have an entry $B[T']$ for every subtree T' of the input tree T in the DP table; entry $B[T']$ equals the cost of the optimal hub labeling for T'; its value can be computed from the values $B[T'']$ of subtrees T'' of T'. We then modify the DP and obtain a polynomial-time approximation scheme (PTAS). We do that by restricting the DP table to subtrees T' with a "small boundary", proving that the obtained algorithm finds a $(1+\varepsilon)$-approximate solution and bounding its running time. We also describe a quasi-polynomial time algorithm, based on the same DP approach. We only present the algorithm for HL_1 and defer the presentation and analysis of a slightly more involved algorithm that works for any HL_p, $p \in [1, +\infty]$, to the full version of the paper. In Section 5.2, we analyze the heuristic for trees proposed by Peleg [24]. Using the primal-dual technique, we show that the heuristic finds a 2-approximation for HL_1. We also give an example where the gap between the optimal solution and the solution that the heuristic finds is $3/2 - \varepsilon$. (For $p > 1$, the gap between the optimal solution for HL_p and the solution that the heuristic finds can be at least $\Omega(p/\log p)$.)

Finally, in Section 6, we prove an $\Omega(\log n)$-hardness for HL_1 and HL_∞ by constructing a reduction from Set Cover. Let us very informally describe the intuition behind the proof for HL_1 (the proof for HL_∞ is similar). Given a Set Cover instance, we construct a graph of constant diameter. In this graph, we have a special vertex r, vertices $S_1, \ldots, S_m$, and vertices $x_1, \ldots, x_n$ (as well as auxiliary vertices). Vertices $S_1, \ldots, S_m$ correspond to the sets, and vertices $x_1, \ldots, x_n$ correspond to the elements of the universe in the Set Cover instance. We design the HL instance in such a way that we only have to satisfy the covering property for pairs (r, x_i) (satisfying other pairs of vertices requires very few hubs); furthermore, the cost of the solution is approximately equal to the size of H_r (to guarantee that, we have many copies of r in the graph; their total contribution to the objective is much greater than the contribution of other vertices). There are multiple shortest paths between r and each x_i: if $x_i \in S_j$ in the Set Cover instance, then there is a path $r \to S_j \to x_i$. In order to cover the pair (r, x_i) we have to choose one of the paths between r and x_i, then choose a vertex w on it, and add it to H_r. The instance is designed in such a way, that if we choose path $r \to S_j \to x_i$, then the "optimal choice" of w is S_j (in particular, we ensure that all x_i cannot choose r as their hub; we do so by having many copies of vertex r and using auxiliary vertices). Therefore, to satisfy all pairs (r, x_i), we have to find the smallest possible number of subsets S_j that cover vertices $x_1, \ldots, x_n$ and add them to H_r; that is, we need to solve the Set Cover instance.

Note: Many of our results can easily be extended to the case of directed graphs (namely, the $O_p(\log D)$-approximation for graphs with unique shortest paths and the $\Omega(\log n)$-hardness results). More details are deferred to the full version of the paper.

2 Preliminaries

2.1 Definitions We will say that a graph $G = (V, E)$ with non-negative edge lengths $l(e)$ has unique shortest paths if there is a unique shortest path between every pair of vertices. (We note that if the lengths of the edges are obtained by measurements, affected by some noise, the graph will satisfy the unique shortest path property with probability 1.)

Recall that the shortest-path diameter D of a graph G is the maximum hop length of a shortest path in G (that is, it is the minimum number D such that every shortest path contains at most D edges). Note that D is upper bounded by the aspect ratio ρ of the graph:

$$D \le \rho \equiv \frac{\max_{u,v \in V} d(u,v)}{\min_{(u,v) \in E} l(u,v)}.$$

Here, $d(u, v)$ is the shortest-path distance in G w.r.t.

edge lengths $l(e)$. In particular, if all edges in G have length at least 1, then $D \leq \text{diam}(G)$, where $\text{diam}(G) = \max_{u,v \in V} d(u,v)$.

We will use the following observation about hub labelings: the covering property for the pair (u,u) requires that $u \in H_u$.

2.2 Linear programming relaxation In this section, we introduce a natural LP formulation of HL_1. Let I be the set of all (unordered) pairs of vertices, including pairs (u,u), which we also denote as $\{u,u\}$, $u \in V$. We use indicator variables x_{uv}, for all $(u,v) \in V \times V$, that represent whether $v \in H_u$ or not. Let $S_{uv} (\equiv S_{vu})$ be the set of all vertices that appear in any of the (possibly many) shortest paths between u and v (including the endpoints u and v). Note that, although the number of shortest paths between u and v might, in general, be exponential in n, the set S_{uv} can always be computed in polynomial time. In case there is a unique shortest path between u and v, we use both S_{uv} and P_{uv} to denote the vertices of that unique shortest path. The constraint implied by the covering property is "$\sum_{w \in S_{uv}} \min\{x_{uw}, x_{vw}\} \geq 1$, for all $\{u,v\} \in I$". The resulting LP relaxation is given below.

$(\mathbf{LP_1})$

$$\begin{aligned} \min: \quad & \sum_{u \in V} \sum_{v \in V} x_{uv} \\ \text{s.t.:} \quad & \sum_{w \in S_{uv}} \min\{x_{uw}, x_{vw}\} \geq 1, \quad \forall \{u,v\} \in I, \\ & x_{uv} \geq 0, \qquad\qquad\qquad \forall (u,v) \in V \times V. \end{aligned}$$

We note that the constraint $\sum_{w \in S_{uv}} \min\{x_{uw}, x_{vw}\} \geq 1$ can be equivalently rewritten as follows: $\sum_{w \in S_{uv}} y_{uvw} \geq 1$, $x_{uw} \geq y_{uvw}$ and $x_{vw} \geq y_{uvw}$. Observe that the total number of variables and constraints remains polynomial, and thus, an optimal solution can always be found efficiently.

One indication that the above LP is indeed an appropriate relaxation of HL is that we can reproduce the result of [15] and get an $O(\log n)$-approximation algorithm for HL by using a very simple rounding scheme. But, we will use the above LP in more refined ways, mainly in conjunction with the notion of pre-hubs, which we introduce later on.

2.3 Hierarchical hub labeling We now define and discuss the notion of hierarchical hub labeling (HHL), introduced by Abraham et al. [5]. The presentation in this section follows closely the one in [5].

DEFINITION 2.1. *Consider a set system $\{H_u\}_{u \in V}$. Let us say that $v \preceq u$ if $v \in H_u$. The set system $\{H_u\}_{u \in V}$ is a hierarchical hub labeling if it is a hub labeling, and $\preceq$ is a partial order.*

We will say that u is higher ranked than v if $u \preceq v$. Every two vertices u and v have a common hub $w \in H_u \cap H_v$, and thus there is a vertex w such that $w \preceq u$ and $w \preceq v$. Therefore, there is the highest ranked vertex in G.

We now define a special type of hierarchical hub labelings. Given a total order $\pi : [n] \to V$, a *canonical* labeling is the hub labeling H that is obtained as follows: $v \in H_u$ if and only if $\pi^{-1}(v) \leq \pi^{-1}(w)$ for all $w \in S_{uv}$. It is easy to see that a canonical labeling is a feasible hierarchical hub labeling. We say that a hierarchical hub labeling H respects a total order π if the implied (by H) partial order is consistent with π. Observe that there might be many different total orders that H respects. In [5], it is proved that all total orders that H respects have the same canonical labeling H', and H' is a subset of H. Therefore, H' is a minimal hierarchical hub labeling that respects the partial order that H implies.

From now on, all hierarchical hub labelings we consider will be canonical hub labelings. Any canonical hub labeling can be obtained by the following process [5]. Start with empty sets H_u, choose a vertex u_1 and add it to each hub set H_u. Then, choose another vertex u_2. Consider all pairs u and v that currently do not have a common hub, but such that u_2 lies on a shortest path between u and v. Add u_2 to H_u and H_v. Then, choose $u_3, \ldots, u_n$, and perform the same step. We get a hierarchical hub labeling. (The hub labeling, of course, depends on the order in which we choose vertices of G.)

This procedure is particularly simple if the input graph is a tree (we will use this in Section 5). In a tree, we choose a vertex u_1 and add it to each hub set H_u. We remove u_1 from the tree and recursively process each connected component of $G - u_1$. No matter how we choose vertices $u_1, \ldots, u_n$, we get a canonical hierarchical hub labeling; given a hierarchical hub labeling H, in order to get a canonical hub labeling H', we need to choose the vertex u_i of highest rank in T' (w.r.t. to the order $\preceq$ defined by H) when our recursive procedure processes subinstance T'. A canonical hub labeling gives a recursive decomposition of the tree to subproblems of gradually smaller size.

3 Pre-hub labeling

We introduce the notion of a pre-hub labeling that we will use in designing algorithms for HL.

DEFINITION 3.1. (PRE-HUB LABELING) *Consider a graph $G = (V,E)$ and a length function $l : E \to \mathbb{R}^+$;*

assume that all shortest paths are unique. A family of sets $\{\widehat{H}_u\}_{u\in V}$, with $\widehat{H}_u \subseteq V$, is called a pre-hub labeling, if for every pair $\{u,v\}$, there exist $u' \in \widehat{H}_u \cap P_{uv}$ and $v' \in \widehat{H}_v \cap P_{uv}$ such that $u' \in P_{v'v}$; that is, vertices u, v, u', and v' appear in the following order along P_{uv}: u, v', u', v (possibly, some of the adjacent, with respect to this order, vertices coincide).

Observe that any feasible HL is a valid pre-hub labeling. We now show how to find a pre-hub labeling given a feasible LP solution.

LEMMA 3.1. *Consider a graph $G=(V,E)$ and a length function $l : E \to \mathbb{R}^+$; assume that all shortest paths are unique. Let $\{x_{uv}\}$ be a feasible solution to* $\mathbf{LP_1}$. *Then, there exists a pre-hub labeling $\{\widehat{H}_u\}_{u\in V}$ such that $|\widehat{H}_u| \le 2\sum_{v\in V} x_{uv}$. In particular, if $\{x_{uv}\}$ is an optimal LP solution and OPT is the ℓ_1-cost of the optimal hub labeling (for HL$_1$), then $\sum_{u\in V} |\widehat{H}_u| \le 2\,OPT$. Furthermore, the pre-hub labeling $\{\widehat{H}_u\}_{u\in V}$ can be constructed efficiently given the LP solution $\{x_{uv}\}$.*

Proof. Let us fix a vertex u. We build the breadth-first search tree T_u (w.r.t. edge lengths; i.e. the shortest-path tree) from u; tree T_u is rooted at u and contains those edges $e \in E$ that appear on a shortest path between u and some vertex $v \in V$. Observe that T_u is indeed a tree and is uniquely defined, since we have assumed that shortest paths in G are unique. For every vertex v, let T'_{uv} be the subtree of T_u rooted at vertex v. Given a feasible LP solution $\{x_{uv}\}$, we define the weight of T'_{uv} to be $\mathcal{W}(T'_{uv}) = \sum_{w\in T'_{uv}} x_{uw}$.

We now use the following procedure to construct set $\widehat{H}_u$. We process the tree T_u bottom up (i.e. we process a vertex v after we have processed all other vertices in the subtree rooted at v), and whenever we detect a subtree T'_{uv} of T_u such that $\mathcal{W}(T'_{uv}) \ge 1/2$, we add vertex v to the set $\widehat{H}_u$. We then set $x_{uw}=0$ for all $w \in T'_{uv}$, and continue (with the updated x_{uw} values) until we reach the root u of T_u. Observe that every time we add one vertex to $\widehat{H}_u$, we decrease the value of $\sum_{v\in V} x_{uv}$ by at least $1/2$. Therefore, $|\widehat{H}_u| \le 2\cdot\sum_{v\in V} x_{uv}$. We will now show that sets $\{\widehat{H}_u\}$ form a pre-hub labeling. To this end, we prove the following two claims.

CLAIM 3.1. *Consider a vertex u and two vertices v_1, v_2 such that $v_1 \in P_{uv_2}$. If $\widehat{H}_u \cap P_{v_1v_2} = \varnothing$, then $\sum_{w\in P_{v_1v_2}} x_{uw} < 1/2$.*

Proof. Consider the execution of the algorithm that defined $\widehat{H}_u$. Consider the moment M when we processed vertex v_1. Since we did not add v_1 to $\widehat{H}_u$, we had $\mathcal{W}(T'_{uv_1}) < 1/2$. In particular, since $P_{v_1v_2}$ lies in T'_{uv_1}, we have $\sum_{w\in P_{v_1v_2}} x'_{uw} < 1/2$, where x'_{uw} is the value of x_{uw} at the moment M. Since none of the vertices on the path $P_{v_1v_2}$ were added to $\widehat{H}_u$, none of the variables x_{uw} for $w \in P_{v_1v_2}$ had been set to 0. Therefore, $x'_{uw} = x_{uw}$ (where x_{uw} is the initial value of the variable) for $w \in P_{v_1v_2}$. We conclude that $\sum_{w\in P_{v_1v_2}} x_{uw} < 1/2$, as required. □

CLAIM 3.2. *For any shortest path P_{uv}, let $u' \in \widehat{H}_u \cap P_{uv}$ be the vertex closest to v among all vertices in $\widehat{H}_u \cap P_{uv}$ and $v' \in \widehat{H}_v \cap P_{uv}$ be the vertex closest to u among all vertices in $\widehat{H}_v \cap P_{uv}$. Then $u' \in P_{v'v}$. (Note that $\widehat{H}_u \cap P_{uv} \ne \varnothing$, since always $x_{uu}=1$ and hence $u \in \widehat{H}_u \cap P_{uv}$; similarly, $\widehat{H}_v \cap P_{uv} \ne \varnothing$.)*

Proof. Let us assume that this is not the case; that is, $u' \notin P_{v'v}$. Then $v' \ne u$ and $u' \ne v$ (otherwise, we would trivially have $u' \in P_{v'v}$). Let u'' be the first vertex after u' on the path $P_{u'v}$, and v'' be the first vertex after v' on the path $P_{v'u}$. Since $u' \notin P_{v'v}$, every vertex of P_{uv} lies either on $P_{v''u}$ or $P_{u''v}$, or both (i.e. $P_{v''u} \cup P_{u''v} = P_{uv}$).

By our choice of u', there are no pre-hubs for u on $P_{u''v}$. By Claim 3.1, $\sum_{w\in P_{u''v}} x_{uw} < 1/2$. Similarly, $\sum_{w\in P_{v''u}} x_{vw} < 1/2$. Thus, $1 > \sum_{w\in P_{uv''}} x_{vw} + \sum_{w\in P_{u''v}} x_{uw} \ge \sum_{w\in P_{uv}} \min\{x_{uw}, x_{vw}\}$. We get a contradiction since $\{x_{uv}\}$ is a feasible LP solution. □

Claim 3.2 shows that $\{\widehat{H}_u\}$ is a valid pre-hub labeling.□

4 Hub labeling on graphs with unique shortest paths

In this section, we present an $O(\log D)$-approximation algorithm for HL$_p$ on graphs with unique shortest paths, where D is the shortest-path diameter of the graph. The algorithm works for every fixed $p \in [1,\infty)$ (the hidden constant factor in the approximation factor $O(\log D)$ depends on p). To simplify the exposition, we present the algorithm for HL$_1$ in this section, and the algorithm for HL$_p$, for arbitrary fixed $p \ge 1$, in Appendix A.

ALGORITHM 4.1. Algorithm for HL$_1$ on graphs with unique shortest paths

1. Solve $\mathbf{LP_1}$ and get an optimal solution x.
2. Obtain a set of pre-hubs $\{\widehat{H}_u\}_{u\in V}$ from x as described in Lemma 3.1.
3. Generate a random permutation $\pi : [n] \to V$ of the vertices.
4. Set $H_u = \varnothing$, for every $u \in V$.
5. **For** ($i = 1$ **to** n) **{**
  For (every $u \in V$**) {**
    For (every $u' \in \widehat{H}_u$**, such that** $\pi_i \in P_{uu'}$ **and** $P_{\pi_i u'} \cap \widehat{H}_u = \{u'\}$)
      If $P_{\pi_i u'} \cap H_u = \varnothing$ **then** $H_u := H_u \cup \{\pi_i\}$
  }

}
6. Return $\{H_u\}_{u \in V}$.

Consider Algorithm 4.1, as given above. The algorithm solves the LP relaxation and computes a pre-hub labeling $\{\widehat{H}_u\}_{u \in V}$ as described in Lemma 3.1. Then it chooses a random permutation π of V and goes over all vertices one-by-one in the order specified by π: π_1, π_2,..., π_n. It adds π_i to H_u if there is a pre-hub $u' \in \widehat{H}_u$ such that the following conditions hold: π_i lies on the path $P_{uu'}$, there are no pre-hubs for u between π_i and u' (other than u'), and currently there are no hubs for u between π_i and u'.

THEOREM 4.1. *Algorithm 4.1 always returns a feasible hub labeling H. The cost of the hub labeling is $\mathbb{E}[\sum_u |H_u|] = O(\log D) \cdot OPT_{LP_1}$ in expectation, where OPT_{LP_1} is the optimal value of* $\mathbf{LP_1}$.

REMARK 4.1. *Note that Algorithm 4.1 can be easily derandomized using the method of conditional expectations: instead of choosing a random permutation π, we first choose $\pi_1 \in V$, then $\pi_2 \in V \setminus \{\pi_1\}$ and so on; each time we choose $\pi_i \in V \setminus \{\pi_1, \ldots, \pi_{i-1}\}$ so as to minimize the conditional expectation $\mathbb{E}[\sum_u |H_u| \,|\, \pi_1, \ldots, \pi_i]$.*

Proof. We first show that the algorithm always finds a feasible hub labeling. Consider a pair of vertices u and v. We need to show that they have a common hub on P_{uv}. The statement is true if $u = v$ since $u \in \widehat{H}_u$ and thus $u \in H_u$. So, we assume that $u \neq v$. Consider the path P_{uv}. Because of the pre-hub property, there exist $u' \in \widehat{H}_u$ and $v' \in \widehat{H}_v$ such that $u' \in P_{v'v}$. In fact, there may be several possible ways to choose such u' and v'. Let us choose u' and v' so that $\widehat{H}_u \cap (P_{u'v'} \setminus \{u', v'\}) = \widehat{H}_v \cap (P_{u'v'} \setminus \{u', v'\}) = \varnothing$ (for instance, choose the closest pair of u' and v' among all possible pairs). Consider the first iteration i of the algorithm such that $\pi_i \in P_{u'v'}$. We claim that the algorithm adds π_i to both H_u and H_v. Indeed, let us verify that the algorithm adds π_i to H_u. We have: (i) π_i lies on $P_{v'u'} \subset P_{uu'}$, (ii) there are no pre-hubs for u on $P_{v'u'} \supset P_{\pi_i u'}$ other than u', (iii) π_i is the first vertex we process on the path $P_{u'v'}$, thus currently there are no hubs on $P_{u'v'}$. Therefore, the algorithm adds π_i to H_u. Similarly, the algorithm adds π_i to H_v.

Now we upper bound the expected cost of the solution. We will charge every hub that we add to H_u to a pre-hub in $\widehat{H}_u$; namely, when we add π_i to H_u, we charge it to pre-hub u'. For every vertex u, we have $|\widehat{H}_u| \leq 2 \sum_w x_{uw}$. We are going to show that every $u' \in \widehat{H}_u$ is charged at most $O(\log D)$ times in expectation. Therefore, the expected number of hubs in H_u is at most $O(2 \sum_w x_{uw} \times \log D)$.

Consider a vertex u and a pre-hub $u' \in \widehat{H}_u$ ($u' \neq u$). Let $u'' \in \widehat{H}_u$ be the closest pre-hub to u' on the path $P_{u'u}$. Observe that all hubs charged to u' lie on the path $P_{u''u'} \setminus \{u''\}$. Let $k = |P_{u''u'} \setminus \{u''\}|$. Note that $k \leq D$. Consider the order $\sigma : [k] \to P_{u''u'} \setminus \{u''\}$ in which the vertices of $P_{u''u'} \setminus \{u''\}$ were processed by the algorithm (σ is a random permutation). Note that σ_i charges u' if and only if σ_i is closer to u' than $\sigma_1, \ldots, \sigma_{i-1}$. The probability of this event is $1/i$. We get that the number of hubs charged to u' is $\sum_{i=1}^{k} \frac{1}{i} = \log k + O(1)$, in expectation. Hence, $\mathbb{E}\left[\sum_{u \in V} |H_u|\right] \leq 2 (\log D + O(1)) \cdot OPT_{LP_1}$. □

5 Hub labeling on trees

In this section, we study the Hub Labeling problem on trees. Observe that if the graph is a tree, the length function l does not play any role in the task of choosing the optimal hubs (it only affects the actual distances between the vertices), and so we assume that we are simply given an unweighted tree $T = (V, E)$, $|V| = n$. We start with proving a structural result about optimal solutions in trees; we show that there always exists a *hierarchical hub labeling* that is also an optimal hub labeling. We then analyze a simple and fast heuristic for HL on trees proposed by Peleg [24], and prove that it gives a 2-approximation for HL_1. We do not know if our analysis is tight, but we prove that there are instances where the heuristic finds a suboptimal solution of cost at least $(\frac{3}{2} - \varepsilon)OPT$ (for every $\varepsilon > 0$). Finally, we present a polynomial-time approximation scheme (PTAS) and a quasi-polynomial-time exact algorithm for HL_1 on trees. We then modify our approach in order to obtain a PTAS and a quasi-polynomial-time exact algorithm for HL_p on trees for any $p \in [1, \infty]$, thus providing a thorough algorithmic understanding of HL, under any cost function, on trees. For simplicity, we only present our results for HL_1, and defer the results for HL_p, $p \in [1, +\infty]$, to the full version of the paper.

5.1 Optimal solutions for trees are hierarchical

We show that any feasible hub labeling solution H can be converted to a hierarchical hub labeling H' of at most the same ℓ_p-cost (for every $p \in [1, \infty]$). Therefore, there always exists an optimal solution that is hierarchical.

THEOREM 5.1. *For every tree $T = (V, E)$, there always exists an optimal HL_p solution that is hierarchical, for every $p \in [1, \infty]$.*

Proof. To prove this, we consider a feasible solution H and convert it to a hierarchical solution H' such that $|H'_u| \leq |H_u|$ for every u. In particular, the ℓ_p-cost of H' is at most the ℓ_p-cost of H for every p.

The construction is recursive (the underlying inductive hypothesis for smaller subinstances being that a feasible HL H can be converted to a hierarchical solution H' such that $|H'_u| \leq |H_u|$ for every u.) First, for each $u \in V$, define an induced subtree $T_u \subseteq T$ as follows: T_u is the union of paths P_{uv} over all $v \in H_u$. In other words, a vertex w belongs to H_u if there is a hub $v \in H_u$ such that $w \in P_{uv}$. Note that T_u is a connected subtree of T.

The crucial property that we need is that $T_u \cap T_v \neq \varnothing$, for every $u, v \in V$. To see this, consider any pair $\{u, v\}$, $u \neq v$. We know that $H_u \cap H_v \cap P_{uv} \neq \varnothing$. Let $w \in H_u \cap H_v \cap P_{uv}$. By construction, $w \in T_u$ and $w \in T_v$, and so $T_u \cap T_v \neq \varnothing$. We now use the fact that a family of subtrees of a tree satisfies the *Helly property* (which first appeared as a consequence of the work of Gilmore [19], and more explicitly a few years later in [21]): suppose that we are given a family of subtrees of T such that every two subtrees in the family intersect, then all subtrees in the family intersect (i.e. they share a common vertex).

Let $r \in \bigcap_{u \in V} T_u$. We remove r from T. Consider the connected components of $T - r$: $Q_1, \ldots, Q_c$. Denote the connected component that contains vertex u by Q^u. Let $\widetilde{H}_u = H_u \cap Q^u$. Note that $|\widetilde{H}_u| \leq |H_u| - 1$, since $r \in T_u$, which, by the definition of T_u, implies that there exists some $w \notin Q^u$ with $w \in H_u$. Consider $u, v \in Q_i$. They have a common hub $w \in H_u \cap H_v \cap P_{uv}$. Since $P_{uv} \subset Q^u = Q^v = Q_i$, we have $w \in \widetilde{H}_u \cap \widetilde{H}_v \cap P_{uv}$. Therefore, $\{\widetilde{H}_u : u \in Q_i\}$ is a feasible hub labeling for Q_i. Now, we recursively find hierarchical hub labelings for $Q_1, \ldots, Q_c$. Denote the hierarchical hub labeling for u in Q^u by H''_u. The inductive hypothesis ensures that $|H''_u| \leq |\widetilde{H}_u| \leq |H_u| - 1$.

Finally, define $H'_u = H''_u \cup \{r\}$, for $u \neq r$, and $H'_r = \{r\}$. We show that H'_u is a hub labeling. Consider $u, v \in V$. If $u, v \in Q_i$ for some i, then $H'_u \cap H'_v \cap P_{uv} \supset H''_u \cap H''_v \cap P_{uv} \neq \varnothing$ since H'' is a hub labeling for Q_i. If $u \in Q_i$ and $v \in Q_j$ $(i \neq j)$, then $r \in H'_u \cap H'_v \cap P_{uv}$. Also, if either $u = r$ or $v = r$, then again $r \in H'_u \cap H'_v \cap P_{uv}$. We conclude that H' is a hub labeling. Furthermore, H' is a *hierarchical* hub labeling: $r \preceq u$ for every u and $\preceq$ is a partial order on every set Q_i; elements from different sets Q_i and Q_j are not comparable w.r.t. $\preceq$.

We have $|H'_u| = |H''_u| + 1 \leq |H_u|$ for $u \neq r$ and $|H'_r| = 1 \leq |H_r|$, as required. □

This theorem allows us to restrict our attention only to hierarchical hub labelings, which have a much simpler structure than arbitrary hub labelings, when we design algorithms for HL on trees.

5.2 An analysis of Peleg's heuristic for $\mathbf{HL}_1$ on trees In this section, we analyze a purely combinatorial algorithm for HL proposed by Peleg in [24] and show that it returns a hierarchical 2-approximate hub labeling on trees (see Algorithm 5.1). (Peleg's result [24], expressed in the hub labeling context, shows that the algorithm returns a hub labeling H with $\max_{u \in V} |H_u| = O(\log n)$ for a tree on n vertices.)

DEFINITION 5.1. *Consider a tree T on n vertices. We say that a vertex u is a balanced separator vertex if every connected component of $T - u$ has at most $n/2$ vertices. The weighted balanced separator vertex for a vertex-weighted tree is defined analogously.*

It is well-known that every tree T has a balanced separator vertex (in fact, a tree may have either exactly one or exactly two balanced separator vertices) and such a separator vertex can be found efficiently (i.e. in linear time) given T. The algorithm by Peleg, named here the Tree Algorithm, is described below (Algorithm 5.1).

ALGORITHM 5.1. Tree Algorithm

Input: a tree T'
Output: a hub labeling H for T'

1. Find a balanced separator vertex r in T'.
2. Remove r and recursively find a HL in each subtree T_i of $T' - r$. Let H' be the labeling obtained by the recursive procedure. (If T' consists of a single vertex, the algorithm does not make any recursive calls.)
3. Return $H_u := H'_u \cup \{r\}$, for every vertex u in $T' - \{r\}$, and $H_r = \{r\}$.

It is easy to see that the algorithm always returns a feasible hierarchical hub labeling, in total time $O(n \log n)$. To bound its cost, we use the primal–dual approach. We consider the dual of $\mathbf{LP_1}$. Then, we define a dual feasible solution whose cost is at least half of the cost of the solution that the algorithm returns. We formally prove the following theorem.

THEOREM 5.2. *The Tree Algorithm is a 2-approximation algorithm for HL_1 on trees.*

Proof. The primal and dual linear programs for HL on trees are given in Figure 1. We note that the dual variables $\{a_{uv}\}_{u,v}$ correspond to unordered pairs $\{u, v\} \in I$, while the variables $\{\beta_{uvw}\}_{u,v,w}$ correspond to ordered pairs $(u, v) \in V \times V$, i.e. β_{uvw} and β_{vuw} are different variables.

As already mentioned, it is straightforward that the algorithm finds a feasible hierarchical hub labeling. We now bound the cost of the solution by constructing

a fractional solution for the DUAL-LP. To this end, we track the execution of the algorithm and gradually define the fractional solution. Consider one iteration of the algorithm in which the algorithm processes a tree T' (T' is a subtree of T). Let r be the balanced separator vertex that the algorithm finds in line 1. At this iteration, we assign dual variables a_{uv} and β_{uvw} for those pairs u and v in T' for which P_{uv} contains vertex r. Let n' be the size of T', $A = 2/n'$ and $B = 1/n'$. Denote the connected components of $T' - r$ by $T_1, ..., T_t$; each T_i is a subtree of T'.

(**PRIMAL-LP**)

$$\begin{aligned}
\min: \quad & \sum_{u\in V}\sum_{v\in V} x_{uv} & \\
\text{s.t.:} \quad & \sum_{w\in P_{uv}} y_{uvw} \geq 1, & \forall\{u,v\}\in I\\
& x_{uw} \geq y_{uvw}, & \forall\{u,v\}\in I,\ \forall w\in P_{uv}\\
& x_{vw} \geq y_{uvw}, & \forall\{u,v\}\in I,\ \forall w\in P_{uv}\\
& x_{uv} \geq 0, & \forall\{u,v\}\in V\times V\\
& y_{uvw} \geq 0, & \forall\{u,v\}\in I,\ \forall w\in P_{uv}
\end{aligned}$$

(**DUAL-LP**)
variables: α_{uv} and β_{uvw} for $w \in P_{uv}$

$$\begin{aligned}
\max: \quad & \sum_{\{u,v\}\in I} \alpha_{uv} & \\
\text{s.t.:} \quad & \alpha_{uv} \leq \beta_{uvw} + \beta_{vuw}, & \forall\{u,v\}\in I\,, u\neq v\\
& & \forall w\in P_{uv}\\
& \alpha_{uu} \leq \beta_{uuu}, & \forall u\in V\\
& \sum_{v:w\in P_{uv}} \beta_{uvw} \leq 1, & \forall(u,w)\in V\times V\\
& \alpha_{uv} \geq 0, & \forall\{u,v\}\in I\\
& \beta_{uvw} \geq 0, & \forall\{u,v\}\in I, \forall w\in P_{uv}\\
& \beta_{vuw} \geq 0, & \forall\{u,v\}\in I, \forall w\in P_{uv}
\end{aligned}$$

Figure 1: Primal and Dual LPs for HL on trees

Observe that we assign a value to each a_{uv} and β_{uvw} exactly once. Indeed, since we split u and v at some iteration, we will assign a value to a_{uv} and β_{uvw} at least once. Consider the first iteration in which we assign a value to a_{uv} and β_{uvw}. At this iteration, vertices u and v lie in different subtrees T_i and T_j of T' (or $r \in \{u, v\}$). Therefore, vertices u and v do not lie in the same subtree T'' in the consecutive iterations; consequently, we will not assign new values to a_{uv} and β_{uvw} later.

For $u \in T_i$ and $v \in T_j$ (with $i \neq j$), we define α_{uv} and β_{uvw} as follows

- $\alpha_{uv} = A$,
- For $w \in P_{ur} \setminus \{r\}$: $\beta_{uvw} = 0$ and $\beta_{vuw} = A$.
- For $w \in P_{rv} \setminus \{r\}$: $\beta_{uvw} = A$ and $\beta_{vuw} = 0$.
- For $w = r$: $\beta_{uvr} = \beta_{vur} = B$.

For $u \in T_i$ and $v = r$, we define α_{uv} and β_{uvw} as follows

- $\alpha_{ur} = A$.
- For $w \in P_{ur} \setminus \{r\}$: $\beta_{urw} = 0$ and $\beta_{ruw} = A$.
- For $w = r$: $\beta_{urr} = \beta_{rur} = B$.

Finally, we set $\alpha_{rr} = \beta_{rrr} = B$.

We now show that the obtained solution $\{\alpha, \beta\}$ is a feasible solution for DUAL-LP. Consider the first constraint: $\alpha_{uv} \leq \beta_{uvw} + \beta_{vuw}$. If $u \neq r$ or $v \neq r$, $A = \alpha_{uv} = \beta_{uvw} + \beta_{vuw} = 2B$. The second constraint is satisfied since $\alpha_{rr} = \beta_{rrr}$.

Let us verify now that the third constraint, $\sum_{v:w\in P_{uv}} \beta_{uvw} \leq 1$, is satisfied. Consider a *non-zero* variable β_{uvw} appearing in the sum. Consider the iteration of the algorithm in which we assign β_{uvw} a value. Let r be the balanced separator vertex during this iteration. Then, $r \in P_{uv}$ (otherwise, we would not assign any value to β_{uvw}) and $w \in P_{rv}$. Therefore, $r \in P_{uw}$; that is, the algorithm assigns the value to β_{uvw} in the iteration when it splits u and w (the only iteration when $r \in P_{uw}$). In particular, we assign a value to all non-zero variables β_{uvw} appearing in the constraint in the same iteration of the algorithm. Let us consider this iteration.

If $u \in T_i \cup \{r\}$ and $w \in T_j$, then every v satisfying $w \in P_{uv}$ lies in T_j. For every such v, we have $\beta_{uvw} = A$. Therefore,

$$\sum_{v:w\in P_{uv}} \beta_{uvw} \leq |T_j| \cdot A \leq \frac{n'}{2} \cdot \frac{2}{n'} = 1,$$

as required. If $u \in T_i \cup \{r\}$ and $w = r$, then we have

$$\sum_{v:w\in P_{uv}} \beta_{uvw} = \sum_{v:r\in P_{uv}} \beta_{uvr} = \sum_{v:r\in P_{uv}} B \leq n'B = 1,$$

as required. We have showed that $\{\alpha, \beta\}$ is a feasible solution. Now we prove that its value is at least half of the value of the hub labeling found by the algorithm. Since the value of any feasible solution of DUAL-LP is at most the cost of the optimal hub labeling, this will prove that the algorithm gives a 2-approximation.

We consider one iteration of the algorithm. In this iteration, we add r to the hub set H_u of every vertex $u \in T'$. Thus, we increase the cost of the hub labeling by n'. We are going to show that the dual variables that we set during this iteration contribute at least $n'/2$ to the value of DUAL-LP.

Let $k_i = |T_i| \leq n'/2$, for all $i \in \{1, \ldots, t\}$. We have $\sum_i k_i = n' - 1$. The contribution C of the variables α_{uv} that we set during this iteration to the objective function equals

$$\begin{aligned} C &= \sum_{i<j} \sum_{u \in T_i, v \in T_j} \alpha_{uv} + \sum_i \sum_{u \in T_i} \alpha_{ur} + \alpha_{rr} \\ &= A \sum_{i<j} k_i k_j + A(n'-1) + B \\ &= \frac{2}{n'} \sum_{i<j} k_i k_j + \frac{2n'-1}{n'}. \end{aligned}$$

Now, since $\sum_{j:j \neq i} k_j = (n' - 1 - k_i) \geq (n'-2)/2$, we have

$$\begin{aligned} \frac{2}{n'} \sum_{i<j} k_i k_j = \frac{1}{n'} \sum_{i \neq j} k_i k_j &= \frac{1}{n'} \sum_i k_i \left(\sum_{j:j \neq i} k_j \right) \\ &\geq \frac{n'-2}{2n'} \sum_i k_i = \frac{(n'-1)(n'-2)}{2n'}. \end{aligned}$$

Thus,

$$C \geq \frac{(4n'-2) + (n'^2 - 3n' + 2)}{2n'} = \frac{n'+1}{2}.$$

We proved that $C \geq n'/2$. This finishes the proof. □

Given the simplicity of the Tree Algorithm, it is interesting to understand whether the 2 approximation factor is tight or not. We do not have a matching lower bound, but we show an asymptotic lower bound of 3/2. The instances that give this 3/2 lower bound are the complete binary trees. Formally, we prove the following lemma, whose proof is deferred to the full version of the paper.

LEMMA 5.1. *The approximation factor of the Tree Algorithm is at least $3/2 - \varepsilon$, for every fixed $\varepsilon > 0$.*

REMARK 5.1. *The Tree Algorithm does not find a good approximation for the ℓ_p-cost, when p is large. Let $k > 1$ be an integer. Consider a tree T defined as follows: it consists of a path $a_1, \ldots, a_k$ and leaf vertices connected to the vertices of the path; vertex a_i is connected to $2^{k-i} - 1$ leaves. The tree consists of $n = 2^k - 1$ vertices. It is easy to see that the Tree Algorithm will first choose vertex a_1, then it will process the subtree of T that contains a_k and will choose a_2, then a_3 and so on. Consequently, the hub set H_{a_k} equals $\{a_1, \ldots, a_k\}$ in the obtained hub labeling. The ℓ_p-cost of this hub labeling is greater than k. However, there is a hub labeling $\widetilde{H}$ for the path $a_1, \ldots, a_k$ with $|\widetilde{H}_{a_i}| \leq O(\log k)$, for all $i \in [k]$. This hub labeling can be extended to a hub labeling of T, by letting $\widetilde{H}_l = \widetilde{H}_{a_i} \cup \{l\}$ for each leaf l adjacent to a vertex a_i. Then we still have $|\widetilde{H}_u| \leq O(\log k)$, for any vertex $u \in T$. The ℓ_p-cost of this solution is $O(n^{1/p} \log k)$. Thus, for $k = p$, the gap between the solution H and the optimal solution is at least $\Omega(p/\log p)$. For $p = \infty$, the gap is at least $\Omega(\log n / \log \log n)$, asymptotically.*

5.3 A PTAS for HL on trees We now present a polynomial-time approximation scheme (PTAS) for HL on trees. For simplicity, we only present the algorithm for HL_1, based on dynamic programming (DP). A modified DP approach along similar lines can also be used in the case of HL_p, for any $p \in [1, \infty]$.

Theorem 5.1 shows that we can restrict our attention only to hierarchical hub labelings. That is, we can find an optimal solution by choosing an appropriate vertex r, adding r to every H_u, and then recursively solving HL on each connected component of $T - r$ (see Section 2.3). Of course, we do not know what vertex r we should choose, so to implement this approach, we use dynamic programming (DP). Let us first consider a very basic dynamic programming solution. We store a table B with an entry $B[T']$ for every subtree T' of T. Entry $B[T']$ equals the cost of the optimal hub labeling for tree T'. Now if r is the common hub of all vertices in T', we have

$$B[T'] = |T'| + \sum_{T'' \text{ is a connected component of } T' - r} B[T'']$$

(the term $|T'|$ captures the cost of adding r to each H_u). We obtain the following recurrence formula for the DP:

$$B[T'] = |T'| + \min_{r \in T'} \sum_{T'' \text{ is a connected component of } T' - r} B[T'']. \tag{5.1}$$

The problem with this approach, however, is that a tree may have exponentially many subtrees, which means that the size of the dynamic program and the running time may be exponential.

To work around this, we will instead store $B[T']$ only for some subtrees T', specifically for subtrees with a "small boundary". For each subtree T' of T, we define its boundary $\partial(T')$ as $\partial(T') := \{v \notin T' : \exists u \in T' \text{ with } (u, v) \in E\}$. Consider now a subtree T' of T and its boundary $S = \partial(T')$. Observe that if $|S| \geq 2$, then the set S uniquely identifies the subtree T': T' is the unique connected component of $T - S$ that has all vertices from S on its boundary (every other connected component of $T - S$ has only one vertex from S on its boundary). If $|S| = 1$, that is, $S = \{u\}$ for some $u \in V$,

then it is easy to see that u can serve as a boundary point for $\deg(u)$ different subtrees.

Fix $\varepsilon < 1$. Let $k = 4 \cdot \lceil 1/\varepsilon \rceil$. In our dynamic program, we only consider subtrees T' with $|\partial(T')| \leq k$. Then, the total number of entries is upper bounded by $\sum_{i=2}^{k} \binom{n}{i} + \sum_{u \in V} \deg(u) = O(n^k)$. Note that now we cannot simply use formula (5.1). In fact, if $|\partial(T')| < k$, formula (5.1) is well defined since each connected component T'' of $T' - r$ has boundary of size at most $|\partial(T')| + 1 \leq k$ for any choice of r (since $\partial(T'') \subseteq \partial(T') \cup \{r\}$). However, if $|\partial(T')| = k$, it is possible that $|\partial(T'')| = k + 1$, and formula (5.1) cannot be used. Accordingly, there is no clear way to find the optimal vertex r. Instead, we choose a vertex r_0 such that for every connected component T'' of $T' - r_0$, we have $|\partial(T'')| \leq k/2 + 1$. To prove that such a vertex exists, we consider the tree T' with vertex weights $w(u) = |\{v \in \partial(T') : (u, v) \in E\}|$ and find a balanced separator vertex r_0 for T' w.r.t. weights $w(u)$ (see Definition 5.1). Then, the weight w of every connected component T'' of $T' - r_0$ is at most $k/2$. Thus, $|\partial(T'')| \leq k/2 + 1 < 3k/4 < k$ (we add 1 because $r_0 \in \partial(T'')$).

The above description implies that the only cases where our algorithm does not perform "optimally" are the subproblems T' with $|\partial(T')| = k$. It is also clear that these subproblems cannot occur too often, and more precisely, we can have at most 1 every $k/2$ steps during the recursive decomposition into subproblems. Thus, we will distribute the cost (amortization) of each such non-optimal step that the algorithm makes over the previous $k/4$ steps before it occurs, whenever it occurs, and then show that all subproblems with boundary of size at most $3k/4$ are solved "almost" optimally (more precisely, the solution to such a subproblem is $(1+4/k)$-approximately optimal). This implies that the final solution will also be $(1 + 4/k)$-approximately optimal, since its boundary size is 0.

We now describe our algorithm in more detail. We keep two tables $B[T']$ and $C[T']$. We will define their values so that we can find, using dynamic programming, a hub labeling for T' of cost at most $B[T'] + C[T']$. Informally, the table C can be viewed as some extra budget that we use in order to pay for all the recursive steps with $|\partial(T')| = k$. We let $C[T'] = \max\{0, (|\partial(T')| - 3k/4) \cdot 4|T'|/k\}$ for every T' with $|\partial(T')| \leq k$. We define B (for $|T'| \geq 3$) by the following recurrence (where r_0 is a balanced separator, and "c.c." means connected component):

$$B[T'] = (1 + 4/k) \cdot |T'| + \min_{r \in T'} \sum_{T'' \text{ is a c.c. of } T' - r} B[T''], \quad \text{if } |\partial(T')| < k,$$

$$B[T'] = \sum_{T'' \text{ is a c.c. of } T' - r_0} B[T''], \quad \text{if } |\partial(T')| = k.$$

The base cases of our recursive formulas are when the subtree T' is of size 1 or 2. In this case, we simply set $B[T'] = 1$, if $|T'| = 1$, and $B[T'] = 3$, if $|T'| = 2$.

In order to fill in the table, we generate all possible subsets of size at most k that are candidate boundary sets, and for each such set we look at the resulting subtree, if any, and compute its size. We process subtrees in increasing order of size, which can be easily done if the generated subtrees are kept in buckets according to their size. Overall, the running time will be $n^{O(k)}$.

We will now show that the algorithm has approximation factor $(1 + 4/k)$ for any $k = 4t$, $t \geq 1$. That is, it is a polynomial-time approximation scheme (PTAS).

THEOREM 5.3. *The algorithm is a polynomial-time approximation scheme (PTAS) for HL_1 on trees.*

Proof. We first argue about the approximation guarantee. The argument consists of an induction that incorporates the amortized analysis that was described above. More specifically, we will show that for any subtree T', with $|\partial(T')| \leq k$, the total cost of the algorithm's solution is at most $B[T'] + C[T']$, and $B[T'] \leq \left(1 + \frac{4}{k}\right) \cdot OPT_{T'}$. Then, the total cost of the solution to the original HL instance is at most $B[T] + C[T]$, and, since $C[T] = 0$, we get that the cost is at most $(1 + 4/k) \cdot OPT$.

The induction is on the size of the subtree T'. For $|T'| = 1$ or $|T'| = 2$, the hypothesis holds. Let's assume now that it holds for all trees of size at most $t \geq 2$. We will argue that it then holds for trees T' of size $t + 1$. We distinguish between the cases where $|\partial(T')| < k$ and $|\partial(T')| = k$.

Case $|\partial(T')| < k$: Let $u_0 \in T'$ be the vertex that the algorithm picks and removes. The vertex u_0 is the minimizer of the expression $\min_{r' \in T'} \sum_{T'' \text{ is a c.c. of } T' - r'} B[T'']$, and thus, using the induction hypothesis, we get that the total cost of the solution returned by the algorithm is at most:

$$\begin{aligned}
ALG(T') &\leq |T'| + \sum_{T'' \text{ is c.c. of } T' - u_0} \Big(B[T''] + C[T''] \Big) \\
&\leq |T'| + \sum_{T'' \text{ is c.c. of } T' - u_0} B[T''] \quad + \\
&+ \sum_{T'' \text{ is c.c. of } T' - u_0} \max\{0, (|\partial(T')| + 1 - 3k/4) \cdot 4|T''|/k\} \\
&= |T'| + \sum_{T'' \text{ is c.c. of } T' - u_0} B[T''] \quad + \\
&+ \max\{0, |\partial(T')| + 1 - 3k/4\} \cdot (4/k) \sum_{T'' \text{ is c.c. of } T' - u_0} |T''|
\end{aligned}$$

$$\leq |T'| + \sum_{T'' \text{ is c.c. of } T'-u_0} B[T''] + 4|T'|/k \quad + \\ + \max\{0, |\partial(T')| - 3k/4\} \cdot 4|T'|/k$$
$$\leq (1+4/k) \cdot |T'| + \left(\sum_{T'' \text{ is c.c. of } T'-u_0} B[T''] \right) + C[T']$$
$$= B[T'] + C[T'].$$

We proved the first part. We now have to show that $B[T'] \leq (1+4/k)OPT_{T'}$. Consider an optimal HL for T'. We may assume that it is a hierarchical labeling. Let $r \in T'$ be the vertex with the highest rank in this optimal solution. We have, $OPT_{T'} = |T'| + \sum_{T'' \text{ is c.c. of } T'-r} OPT_{T''}$. By definition, we have that

$$B[T'] = (1+4/k)|T'| + \min_{u \in T'} \sum_{T'' \text{ is c.c. of } T'-u} B[T'']$$
$$\leq (1+4/k)|T'| + \sum_{T'' \text{ is c.c. of } T'-r} B[T'']$$
$$\overset{(ind.hyp.)}{\leq} (1+4/k)|T'| + (1+4/k) \cdot \sum_{T'' \text{ is c.c. of } T'-r} OPT_{T''}$$
$$= (1+4/k) \cdot OPT_{T'}.$$

Case $|\partial(T')| = k$: Using the induction hypothesis, we get that the total cost of the solution returned by the algorithm is at most:

$$ALG(T') \leq |T'| + \sum_{T'' \text{ is c.c. of } T'-r_0} B[T''] \quad + \\ + \sum_{T'' \text{ is c.c. of } T'-r_0} C[T''].$$

By our choice of r_0, we have $|\partial(T'')| \leq 3k/4$, and so $C[T''] = 0$, for all trees T'' of the forest $T' - r_0$. Thus,

$$ALG(T') \leq |T'| + \sum_{T'' \text{ is c.c. of } T'-r_0} B[T'']$$
$$= C[T'] + B[T'].$$

We now need to prove that $B[T'] \leq (1+4/k) \cdot OPT_{T'}$. We have,

$$B[T'] = \sum_{T'' \text{ is c.c. of } T'-r_0} B[T'']$$
$$\overset{(ind.hyp.)}{\leq} \sum_{T'' \text{ is c.c. of } T'-r_0} \left(1+\frac{4}{k}\right) OPT_{T''}$$
$$\leq \left(1+\frac{4}{k}\right) OPT_{T'},$$

where in the the last inequality we use that $\sum_{T''} OPT_{T''} \leq OPT_{T'}$, which can be proved as follows. We convert the optimal hub labeling H' for T' to a set of hub labelings for all subtrees T'' of T': the hub labeling H'' for T'' is the restriction of H' to T''; namely, $H''_v = H'_v \cap V(T'')$ for every vertex $v \in T''$; it is clear that the total number of hubs in labelings H'' for all subtrees T'' is at most the cost of H'. Also, the cost of each hub labeling H'' is at least $OPT_{T''}$. The inequality follows.

We have considered both cases, $|S| < k$ and $|S| = k$, and thus shown that the hypothesis holds for any subtree T' of T. In particular, it holds for T. Therefore, the algorithm finds a solution of cost at most $B[T] + C[T] = B[T] \leq \left(1 + \frac{4}{k}\right) OPT$.

Setting $k = 4 \cdot \lceil 1/\varepsilon \rceil$, as already mentioned, we get a $(1+\varepsilon)$-approximation, for any fixed $\varepsilon \in (0,1)$, and the running time of the algorithm is $n^{O(1/\varepsilon)}$. □

5.4 A quasi-polynomial time algorithm for HL on trees The dynamic programming approach of the previous section can also be used to obtain a quasi-polynomial time algorithm for HL_p on trees. This is done by observing that the set of boundary vertices of a subtree is a subset of the hub set of every vertex in that subtree, and by proving that in an optimal solution for HL_p, all vertices have hub sets of size at most $O(\log n)$, for $p \in \{1, \infty\}$, and of size at most $O(\log^2 n)$ in the case of $p \in (1, \infty)$. Thus, restricting our DP to subinstances with polylogarithmic boundary size, we are able to get an exact quasi-polynomial time algorithm. Again, we only present the algorithm for HL_1.

In order to get an exact quasi-polynomial time algorithm for HL_1 on trees, we first prove that any optimal hierarchical hub labeling solution H satisfies $\max_{u \in V} |H_u| = O(\log n)$.

THEOREM 5.4. *Let H be an optimal (for HL_1) hierarchical hub labeling for a tree T. Then* $\max_{u \in T} |H_u| = O(\log n)$.

We will need to prove a few auxiliary claims before we proceed with the proof of Theorem 5.4. Consider a canonical hierarchical labeling for T. Recall that we may assume that H is constructed as follows: we choose a vertex r in T of highest rank and add it to each hub set H_v (for v in T), then recursively process each tree in the forest $T - r$. For every vertex u, let T_u be the subtree of T, which we process when we choose $r = u$. (Note that u is the vertex of highest rank in T_u.)

Let us consider the following "move ahead" operation that transforms a hierarchical hub labeling H to a hierarchical hub labeling H'. The operation is defined by two elements $r_1 \prec r_2$ (i.e. $r_2 \in T_{r_1}$) as follows. Consider the process that constructs sets H_u, described in the previous paragraph. Consider the recursive step when we process subtree T_{r_1}. At this step, let us inter-

vene in the process: choose vertex r_2 instead of choosing r_1. Then, when we recursively process a subtree T' of $T_{r_1} - r_2$, we choose the vertex of highest rank in T' (as indicated by H). Clearly, we obtain a hierarchical hub labeling. Denote it by H'.

LEMMA 5.2. *Suppose that we move r_2 ahead of r_1 ($r_1 \prec r_2$) and obtain labeling H'. Let $\widetilde{T}$ be the connected component of $T_{r_1} - r_2$ that contains r_1. The following statements hold.*

1. *If $u \notin T_{r_1}$, then $H'_u = H_u$.*
2. *If $u \in T_{r_1}$, then $H'_u \subset H_u \cup \{r_2\}$.*
3. *If $u \in T_{r_1} \setminus \widetilde{T}$, then $r_1 \notin H'_u$.*

Proof. 1. The "move ahead" operation affects only the processing of the tree T_{r_1} and its subtrees. Therefore, if $u \notin T_{r_1}$, then $H'_u = H_u$.

2. Consider a vertex $v \in H'_u \setminus \{r_2\}$; we will prove that $v \in H_u$. Consider the step when we add v to H'_u. Denote the tree that we process at this step by T'. Then $u \in T'$ and v is the highest ranked element in T'. In particular, $v \prec u$, and, by the definition of the ordering $\preceq$, $v \in H_u$.

3. If $u = r_2$, the statement is trivial. Otherwise, note that the subtree of $T_{r_1} - r_2$ that contains u does not contain r_1 (because r_1 lies in the subtree $\widetilde{T}$), and, thus, u and r_1 never lie in the same subtree during later recursive calls. Therefore, we do not add r_1 to H'_u. □

Consider a canonical hierarchical labeling H, a vertex u and subtree T_u, as defined before. Let T' be the largest connected component of $T_u - u$. We define $M_u = |T_u| - |T'|$.

CLAIM 5.1. *Assume that H is an optimal hierarchical hub labeling for a tree T. Consider two vertices u and v. If $u \prec v$, then $M_u \geq M_v$.*

Proof. Without loss of generality, we can assume that there is no element w between u and v in the partial order $\preceq$ (the result for arbitrary $u \prec v$ will follow by transitivity). Let T' be the largest connected component of $T_u - u$. Consider two cases.

Case $v \notin T'$. Then $M_v \leq |T_v| < |T_u| - |T'|$, since T_v is a proper subset of $T_u \setminus T'$.

Case $v \in T'$. Then $T' = T_v$, and v is the highest ranked vertex in T'. Let T'' be the union of v and all subtrees of $T_u - v$ that do not contain u. Note that $T' - T''$ is a connected component of $T' - v = T_v - v$, and, thus, $M_v \leq |T'| - |T' - T''| \leq |T''|$. Also, note that $v \in H_w$ for $w \in T'$, since v is the highest ranked vertex in T'.

Let us move v ahead of u, and obtain a hub labeling H'. We have, by Lemma 5.2,

1. If $w \notin T_u$, then $H'_w = H_w$; thus, $|H'_w| = |H_w|$.
2. If $w \in T_u - T'$, then $H'_w \subset H_w \cup \{v\}$; thus, $|H'_w| \leq |H_w| + 1$.
3. If $w \in T' - T''$, then $H'_w \subset H_w \cup \{v\} = H_w$; thus, $|H'_w| \leq |H_w|$.
4. If $w \in T''$, then $H'_w \subset H_w$ and $u \in H_w \setminus H'_w$; thus, $|H'_w| \leq |H_w| - 1$.

Since H is an optimal labeling

$$\sum_w |H_w| \leq \sum_w |H'_w|.$$

Therefore, $|T_u - T'| \geq |T''|$. Recall that $M_u = |T_u - T'|$ and $|T''| \geq M_v$. We conclude that $M_u \geq M_v$. □

CLAIM 5.2. *Assume that H is an optimal hierarchical hub labeling for a tree T. For every vertex u, we have $M_u \geq |T_u|/6$.*

Proof. Assume to the contrary that there is a vertex u_1 such that $M_{u_1} < |T_{u_1}|/6$. Denote $n' = |T_{u_1}|$. Let T' be the largest connected component of $T_{u_1} - u_1$, and u_2 be the vertex of highest rank in T'. Let T'' be the largest connected component of $T' - u_2$, and u_3 be the vertex of highest rank in T''. Denote $M_1 = M_{u_1}$ and $M_2 = M_{u_2}$. We have $u_1 \preceq u_2$, and, therefore, $n'/6 > M_1 \geq M_2$. By the definition of T', T'' and numbers M_1, M_2, we have

$$|T'| = |T_{u_1}| - M_1 > \frac{5}{6}n'; \qquad |T''| = |T'| - M_2 > \frac{4}{6}n'.$$

Note that u_1 belongs to n' hub sets H_w (namely, it belongs to H_w for $w \in T_{u_1}$), u_2 belongs to more than $\frac{5}{6}n'$ hub sets H_w, and u_3 belongs to more than $\frac{4}{6}n'$ hub sets H_w.

Let c be balanced separator vertex in T_{u_1}. Clearly, in H we have $u_1 \preceq c$. Consider the hub labeling H' obtained by moving c ahead of u_1. By Lemma 5.2, $H'_w \subset H_w \cup \{c\}$ for $w \in T_{u_1}$, and $H'_w = H_w$ for $w \notin T_{u_1}$. Since every connected component of $T_{u_1} - c$ contains at most $n'/2$ vertices, each of the vertices u_1, u_2, and u_3 will lie in a connected component of size at most $n'/2$. This means that u_1 belongs to at most $n'/2$ hub sets H'_w, and similarly u_2 and u_3 belong to at most $n'/2$ hub sets (each) in H'. We get that

$$\sum_w |H'_w| \leq \Big(\sum_w |H_w|\Big) + \underbrace{n' - 1}_{\substack{\text{bound on cost} \\ \text{of new hub } c}} +$$
$$- \Big(\underbrace{(n' - n'/2) + (5n'/6 - n'/2) + (4n'/6 - n'/2)}_{\substack{\text{difference in number of appearances} \\ \text{of hubs } u_1,u_2,u_3 \text{ in hub labelings } H \text{ and } H'}} \Big)$$
$$< \sum_w |H_w|.$$

We get a contradiction with the optimality of the hub labeling $\{H_w\}$. □

Proof. [Proof of Theorem 5.4] By Claim 5.2, we get that during the decomposition of T into subproblems, the sizes of the subproblems drop by a constant factor. Therefore, the depth of the decomposition tree is $O(\log n)$, and so for every $u \in V$, we must have $|H_u| = O(\log n)$. □

We are now ready to present our quasi-polynomial time algorithm.

THEOREM 5.5. *There exists a DP algorithm that is an exact quasi-polynomial time algorithm for HL_1 on trees, with running time $n^{O(\log n)}$.*

Proof. The DP algorithm is the same as the algorithm presented in Section 5.3, with $k = O(\log n)$, without the balancing steps that occur when the boundary of a subproblem is of size exactly k. The proof relies on the simple observation that the set of boundary vertices that describes a subtree is equal to, or a subset of, the set of hubs that have been assigned so far by an HHL solution to all the vertices of this subtree. In other words, the set of boundary vertices corresponds to some of the vertices of the path in the recursion tree that starts from the root (the vertex with the highest rank in T), and goes down to the internal vertex in the recursion tree that describes the current subproblem. By Theorem 5.4, we know that any such path has at most $k = c \cdot \log n$ vertices, for some constant c. Thus, when looking for an optimal solution, we only have to consider subproblems that can be described by such paths, and so we can store entries in the dynamic program table only for subtrees T' with $|\partial(T')| \leq k = c \cdot \log n$.

This implies that we can use a simple DP algorithm with entries defined by formula (5.1) of Section 5.3 for T' with $|\partial(T')| \leq k$ and $B[T'] = \infty$ for T' with $|\partial(T')| > k$. As we already showed, the DP table has size $n^{O(k)} = n^{O(\log n)}$ (since we don't need to have any entries $B[T']$ for T' with $|\partial(T')| > k$). The running time of the algorithm is $n^{O(\log n)}$. □

6 Hardness of approximating hub labeling on general graphs

In this section, we prove that HL_1 and HL_∞ are NP-hard to approximate on general graphs with multiple shortest paths within a factor better than $\Omega(\log n)$, by using the $\Omega(\log n)$-hardness results for Set Cover. This implies that the current known algorithms for HL_1 and HL_∞ are optimal (up to constant factors).

6.1 An $\Omega(\log n)$-hardness for HL_1 In this section, we show that it is NP-hard to approximate HL_1 on general graphs with multiple shortest paths within a factor better than $\Omega(\log n)$. We will use the hardness results for Set Cover, that, through a series of works spanning more than 20 years [23, 17, 25, 7], culminated in the following theorem.

THEOREM 6.1. (DINUR & STEURER [16]) *For every $\alpha > 0$, it is NP-hard to approximate Set Cover to within a factor $(1-\alpha) \cdot \ln n$, where n is the size of the universe.*

We start with an arbitrary unweighted instance of Set Cover. Let $\mathcal{X} = \{x_1, ..., x_n\}$ be the universe and $\mathcal{S} = \{S_1, ..., S_m\}$ be the family of subsets of $\mathcal{X}$, with $m = \texttt{poly}(n)$. Our goal is to pick the smallest set of indices $I \subseteq [m]$ (i.e. minimize $|I|$) such that $\bigcup_{i \in I} S_i = \mathcal{X}$.

We will construct an instance of HL such that, given an $f(n)$-approximation algorithm for HL_1, we can use it to construct a solution for the original Set Cover instance of cost $O(f(\texttt{poly}(n))) \cdot OPT_{SC}$, where OPT_{SC} is the cost of the optimal Set Cover solution. Formally, we prove the following theorem.

THEOREM 6.2. *Given an arbitrary unweighted Set Cover instance $(\mathcal{X}, \mathcal{S})$, $|\mathcal{X}| = n$, $|\mathcal{S}| = m$, with optimal value OPT_{SC}, and an $f(n)$-approximation algorithm for HL_1, there is an algorithm that returns a solution for the Set Cover instance of cost $O(f(\Theta(\max\{n, m\}^3))) \cdot OPT_{SC}$.*

Using the above theorem, if we assume that $f(n) = o(\log n)$, then $O(f(\Theta(\max\{n, m\}^3))) = O(f(\texttt{poly}(n))) = o(\log \texttt{poly}(n)) = o(\log n)$, and so this would imply that we can get a $o(\log n)$-approximation algorithm for Set Cover. By Theorem 6.1, this is NP-hard, and so we must have $f(n) = \Omega(\log n)$.

COROLLARY 6.1. *It is NP-hard to approximate HL_1 to within a factor $c \cdot \log n$, for some constant c, on general graphs (with multiple shortest paths).*

Before proving Theorem 6.2, we need the following lemma.

LEMMA 6.1. *Let $G = (V, E)$ and $l : E \to \mathbb{R}^+$ be an instance of HL_1, in which there exists at least one vertex $u \in V$ with degree 1, and let w be its unique neighbor. Then, any feasible solution $\{H_v\}_{v \in V}$ can be converted to a solution H' of at most the same cost, with the property that $H'_u = H'_w \cup \{u\}$ and $u \notin H'_v$, for every vertex $v \neq u$.*

Proof. Let $\{H_v\}$ be any feasible solution for HL, and let u be a vertex of degree 1, and w its unique neighbor. If the desired property already holds for every vertex of

degree 1, then we are done. So let us assume that the property does not hold for some vertex u. Let w be its neighbor and

$$B = \begin{cases} H_w \setminus \{u\}, & \text{if } |H_w \setminus \{u\}| \leq |H_u \setminus \{u\}|, \\ H_u \setminus \{u\}, & \text{otherwise.} \end{cases}$$

Let

- $H'_u = B \cup \{u, w\}$.
- $H'_w = B \cup \{w\}$.
- $\forall v \in V \setminus \{u, w\}$,

$$H'_v = \begin{cases} H_v, & \text{if } u \notin H_v, \\ (H_v \setminus \{u\}) \cup \{w\}, & \text{otherwise.} \end{cases}$$

We first check the feasibility of H'. The pairs $\{u, w\}$, and $\{v, v\}$, for all $v \in V$, are clearly satisfied. Also, every pair $\{v, v'\}$ with $v, v' \notin \{u, w\}$ is satisfied, since $u \notin S_{vv'}$. Consider now a pair $\{u, v\}$, with $v \in V \setminus \{u, w\}$. If $u \in H_u \cap H_v$, we have $w \in H'_u \cap H'_v$. Otherwise, $\{u, v\}$ is covered with some vertex $z \in S_{uv} \setminus \{u\}$, and since $S_{uv} \setminus \{u\} = S_{wv}$, we have that $z \in H_u \cap H_v$ and $z \in H_w \cap H_v$. It follows that $z \in H'_u \cap H'_v$. Now, consider a pair $\{w, v\}$, $v \in V \setminus \{u, w\}$. We have either $H'_w = (H_w \setminus \{u\}) \cup \{w\}$, which gives $H'_w \cap S_{wv} = H_w \cap S_{vw}$, or $H'_w = (H_u \setminus \{u\}) \cup \{w\}$. In the latter case, either $u \in H_v$ and so $w \in H'_v$, or $H_u \cap S_{uv} = H_u \cap S_{wv}$. It is easy to see that in all cases the covering property is satisfied.

We now argue about the cost of H'. We distinguish between the two possible values of B:

- $B = H_w \setminus \{u\}$: In this case, $|H'_w| \leq |H_w|$, since $w \in B$. If $u \in H_w$, then $|H'_u| = |H_w| \leq |H_u|$. Otherwise, it holds that $|H_w| \leq |H_u| - 1$, and so $|H'_u| = |H_w| + 1 \leq |H_u|$. For all $v \in V \setminus \{u, w\}$, it is obvious that $|H'_v| \leq |H_v|$.
- $B = H_u \setminus \{u\}$: If $w \in H_u$, then $|H'_w| = |H_u| - 1 < |H_w \setminus \{u\}| \leq |H_w|$, and $|H'_u| = |H_u|$. Otherwise, we must have $u \in H_w$, which means that $|H_u| < |H_w|$. Thus, $|H'_w| = |H_u| < |H_w|$, and $|H'_u| = |H_u| + 1$, which gives $|H'_u| + |H'_w| < |H_u| + 1 + |H_w|$, and so $|H'_u| + |H'_w| \leq |H_u| + |H_w|$. Again, it is obvious that $|H'_v| \leq |H_v|$, for all $v \in V \setminus \{u, w\}$.

Thus, in all cases, H' is a feasible hub labeling that satisfies the desired property and whose cost is $\sum_{v \in V} |H'_v| \leq \sum_{v \in V} |H_v|$. □

We are now ready to prove Theorem 6.2.

Proof. [Proof of Theorem 6.2] Given an unweighted Set Cover instance, we create a graph $G = (V, E)$ and HL_1 instance. We fix two parameters A and B (whose values we specify later) and do the following (see Figure 2):

- The 2 layers directly corresponding to the Set Cover instance are the 3^{rd} and the 4^{th} layer. In the 3^{rd} layer we introduce one vertex for each set $S_i \in \mathcal{S}$, and in the 4^{th} layer we introduce one vertex for each element $x_j \in \mathcal{X}$. We then connect x_j to S_i if and only if $x_j \in S_i$.
- The 2^{nd} layer contains A vertices in total $(\{r_1, ..., r_A\})$, where r_i is connected to every vertex S_j of the 3^{rd} layer.
- For each vertex r_i we introduce B new vertices in the 1^{st} layer, where each such vertex $t_j^{(i)}$ has degree one and is connected to r_i.
- Similarly, for each x_i we introduce B new vertices in the 5^{th} layer, where each such $y_j^{(i)}$ has degree one and is connected to x_i.
- Finally, we introduce a single vertex W in the 6^{th} layer, which is connected to every x_i of the 4^{th} layer.

We also assign lengths to the edges. The (black) edges (W, x_i) have length $\varepsilon < 1/2$ for every $x_i \in \mathcal{X}$, while all other (brown) edges have length 1. We will show that by picking the parameters A and B appropriately, we can get an $O(f(\texttt{poly}(n))$-approximation for the Set Cover instance, given an $f(n)$-approximation for HL_1.

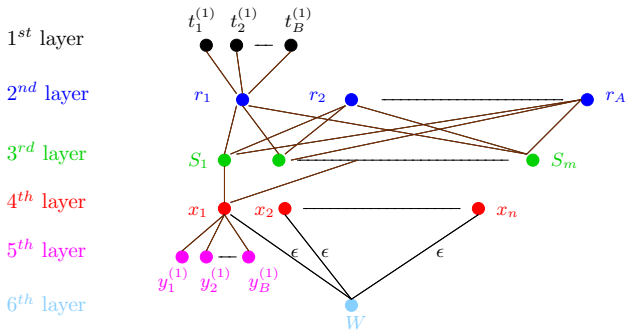


Figure 2: The HL_1 instance corresponding to an arbitrary unweighted Set Cover instance

We will now define a solution for this HL instance, whose cost depends on the cost of the optimal Set Cover. Let $I \subseteq [m]$ be the set of indices of an optimal Set Cover solution. We define the following HL solution, given in the array below. We use the notation $S(x_i)$ to denote an (arbitrarily chosen) set in the optimal Set Cover solution

that covers x_i. We also write $[n] = \{1, ..., n\}$.

Layer	Hubs
1	$\forall i \in [A], j \in [B],\ H_{t_j^{(i)}} = H_{r_i} \cup \{t_j^{(i)}\}$
2	$\forall i \in [A],\ H_{r_i} = \{r_i\} \cup \{S_j : j \in I\}$
3	$\forall i \in [m],\ H_{S_i} = \{S_i, W\} \cup \{r_1, ..., r_A\} \cup \cup\{x_j : x_j \in S_i\}$
4	$\forall i \in [n],\ H_{x_i} = \{x_i, W, S(x_i)\}$
5	$\forall i \in [n], j \in [B],\ H_{y_j^{(i)}} = H_{x_i} \cup \{y_j^{(i)}\}$
6	$H_W = \{W\} \cup \{S_1, ..., S_m\} \cup \{r_1, ..., r_A\}$

We argue that the above solution is a feasible solution for the constructed instance. To this end, we consider all possible pairs of vertices for all layers. The notation "i - j" means that we check a pair with one vertex at layer i and the other at layer j. The pairs $\{u, u\}$ are trivially satisfied, so we will not consider them below:

- 1 - 1: $\{t_a^{(i)}, t_b^{(j)}\}$. If $i = j$, then the pair is covered by r_i. If $i \neq j$, then $H_{t_a^{(i)}}$ and $H_{t_b^{(j)}}$ contain the same sets from the 3^{rd} layer, and so at least one shortest path is covered.
- 1 - 2: $\{t_a^{(i)}, r_j\}$. If $i = j$, then $r_i \in H_{t_a^{(i)}}$. If $i \neq j$, then $H_{t_a^{(i)}}$ and H_{r_j} contain the same sets from the 3^{rd} layer, and so at least one shortest path is covered.
- 1 - 3: $\{t_a^{(i)}, S_j\}$. Observe that $r_i \in H_{S_j} \cap H_{t_a^{(i)}}$, and so the pair is covered.
- 1 - 4: $\{t_a^{(i)}, x_j\}$. The pair is covered by $S(x_j)$.
- 1 - 5: $\{t_a^{(i)}, y_b^{(j)}\}$. Again, the pair is covered by $S(x_j)$.
- 1 - 6: $\{t_a^{(i)}, W\}$. The pair is covered by r_i.
- 2 - 2: Any such pair is covered by any of the S_j's, with $j \in I$.
- 2 - 3: $\{r_i, S_j\}$. We have $r_i \in H_{S_j}$.
- 2 - 4: $\{r_i, x_j\}$. The pair is covered by $S(x_j)$.
- 2 - 5: Since the "2 - 4" pairs are covered, the "2 - 5" pairs are covered as well.
- 2 - 6: Vertex W is directly connected to all vertices of the 2^{nd} layer.
- 3 - 3: Any $\{S_i, S_j\}$ is covered by any vertex of the 2^{nd} layer.
- 3 - 4: $\{S_i, x_j\}$. If $x_j \in S_i$, then $x_j \in H_{S_i}$. If $x_j \notin S_i$, then $W \in H_{S_i} \cap H_{x_j}$.
- 3 - 5: $\{S_i, y_a^{(j)}\}$. If $x_j \in S_i$, then they are covered by x_j. If not, then they are covered by W.
- 3 - 6: There is a direct connection.
- 4 - 4: Any such pair is covered by W.
- 4 - 5: $\{x_i, y_a^{(j)}\}$. If $i = j$, then $x_i \in H_{y_a^{(i)}}$. If not, then they are covered by W.
- 4 - 6: There is a direct connection.
- 5 - 5: $\{y_a^{(i)}, y_b^{(j)}\}$. If $i = j$, then they are covered by x_j. If not, then they are covered by W.
- 5 - 6: There is a direct connection.

Thus, the above solution is indeed a feasible one. We compute its cost (each term from left to right corresponds to the total cost of the vertices of the corresponding layer):

$$\begin{aligned}\text{COST} &\leq A \cdot B \cdot (OPT_{SC} + 2) + A \cdot (OPT_{SC} + 1) + \\ &\quad + m(A + n + 2) + 3n + 4B \cdot n + (1 + m + A) \\ &= O(AB \cdot OPT_{SC}) + O(A \cdot OPT_{SC}) + \\ &\quad + O(Am + mn) + O(n) + O(Bn) + O(m + A).\end{aligned}$$

We set $A = B = \max\{m, n\}^{3/2}$, and let $N = \Theta(A \cdot B)$ denote the number of vertices of the graph. Then, the total cost is dominated by the term $AB \cdot OPT_{SC}$, and so we get that the cost OPT of the optimal HL_1 solution is at most $c \cdot AB \cdot OPT_{SC}$, for some constant c. It is also easy to see that $OPT \geq A \cdot B$.

Assuming that we have an $f(n)$-approximation for HL_1, we can get a solution H' of cost $\|H'\|_1 \leq c \cdot f(N) \cdot AB \cdot OPT_{SC}$. We will show that we can extract a feasible Set Cover solution of cost at most $\frac{\|H'\|_1}{AB}$.

To extract a feasible Set Cover, we first modify H'. Using Lemma 6.1, we can obtain a solution H'' such that for every $t_i^{(j)}$ we have $H''_{t_i^{(j)}} = H''_{r_j} \cup \{t_i^{(j)}\}$ and $t_i^{(j)} \notin H''_v$, for every vertex $v \neq t_i^{(j)}$. We also apply the lemma for the vertices of the 5^{th} layer. In addition to that, we add $\{r_1, ..., r_A\}$ to the hub set of W, increasing the cost by at most A. Thus, we end up with a solution H'' of cost at most $c \cdot f(N) \cdot AB \cdot OPT_{SC} + A \leq c' \cdot f(N) \cdot AB \cdot OPT_{SC}$.

We now look at every vertex r_i of the 2^{nd} layer for which we have $x_j \in H''_{r_i}$, for some $j \in [n]$. This x_j can only be used to connect r_i to x_j and to the $\{y_a^{(j)}\}$'s. In that case, we can remove x_j from H''_{r_i} and all $H''_{t_b^{(i)}}$, and add r_i to H''_{x_j} and to all $H''_{y_a^{(j)}}$. The cost of the solution cannot increase, and we again call this new solution H''.

We are ready to define our Set Cover solution. For each $i \in [A]$, we define $F_i = H''_{r_i} \cap \{S_1, ..., S_m\}$. Let $Z_i = |\mathcal{X} \setminus \bigcup_{j \in F_i} S_j|$ be the number of uncovered elements. If $Z_i = 0$, then F_i is a valid Set Cover. If $Z_i > 0$, then we cover the remaining elements using some extra sets (at most Z_i such sets). At the end, we return $\min_{i \in [A]}\{|F_i| + Z_i\}$.

In order to analyze the cost of the above algorithm, we need the following observation. Let us look at H''_{r_i}, and an element x_j that is not covered. By the structure of H'', this means that $r_i \in H''_{x_j}$. Thus, the number of uncovered elements Z_i contributes a term $(B+1)Z_i$ to the cost of the HL_1 solution. For each i, the number of uncovered elements Z_i implies an increase in the cost of the 4^{th} and 5^{th} layer, which is "disjoint" with the increase for $j \neq i$, and so the total cost of the 1^{st}, 2^{nd}, 4^{th} and 5^{th} layer combined is at least $\sum_{i=1}^{A}(B+1)(|H''_{r_i}| + Z_i)$. So, there must exist an i such that

$$|H''_{r_i}| + Z_i \leq \frac{\|H''\|_1}{A(B+1)}.$$

We pick the Set Cover with cost at most $\min_j\{|F_j| + Z_j\} \leq \min_j\{|H''_{r_j}| + Z_j\}$, and so we end up with a feasible Set Cover solution of cost at most

$$\frac{c' \cdot f(N) \cdot AB \cdot OPT_{SC}}{AB} = c' \cdot f(N) \cdot OPT_{SC}.$$

□

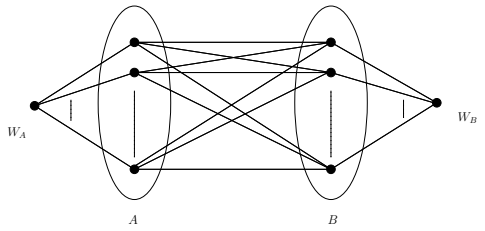
6.2 An $\Omega(\log n)$-hardness for HL_∞ In this section, we will show that it is NP-hard to approximate HL_∞ to within a factor better than $\Omega(\log n)$. We will again use the hardness results for Set Cover.

THEOREM 6.3. *Given an arbitrary unweighted Set Cover instance $(\mathcal{X}, \mathcal{S})$, $|\mathcal{X}| = n$, $|\mathcal{S}| = m = \texttt{poly}(n)$, with optimal value OPT_{SC}, and an $f(n)$-approximation algorithm for HL_∞, there is an algorithm that returns a solution for the Set Cover instance with cost at most $O\left(f(O(n^4 \cdot m))\right) \cdot OPT_{SC}$.*

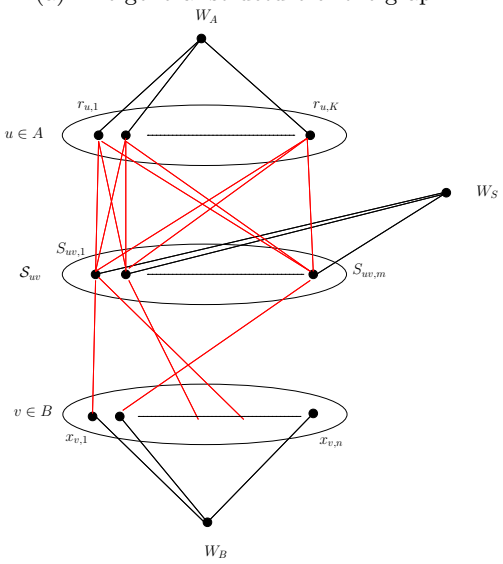
Proof. Let $\mathcal{X} = \{x_1, ..., x_n\}$ and $\mathcal{S} = \{S_1, ..., S_m\}$, $S_i \subseteq \mathcal{X}$ be a Set Cover instance. We will construct an instance of HL_∞, such that, given an $f(n)$-approximation algorithm for it, we will be able to solve the Set Cover instance within a factor of $O\left(f(O(n^4 m))\right)$. We now describe our construction:

- We introduce a complete bipartite graph (A, B, E). By slightly abusing notation, we denote $|A| = A$ and $|B| = B$, where A and B are two parameters to be set later on.
- Each vertex $u \in A$ "contains" K vertices $\{r_{u,1}, ..., r_{u,K}\}$.
- Each vertex vertex $v \in B$ "contains" a copy of the universe $\{x_{v,1}, ..., x_{v,n}\}$.
- Each edge (u, v) is replaced by an intermediate layer of vertices $\mathcal{S}_{uv} = \{S_{uv,1}, ..., S_{uv,m}\}$, which is essentially one copy of $\mathcal{S}$. We then connect every vertex $r_{u,i}$, $i \in [K]$, to every vertex $S_{uv,j}$, $j \in [m]$, and we also connect each $S_{uv,j}$ to $x_{v,t}$, if $x_t \in S_j$. All these edges (colored red in the figure) have length 1.
- Finally, we introduce three extra vertices W_A and W_B and W_S, and the edges $(W_A, r_{u,i})$, for all $u \in A$ and $i \in [K]$, the edges $(W_B, x_{v,j})$, for all $v \in B$ and $j \in [n]$, and the edges $(W_S, S_{uv,j})$, for all $u \in A$, $v \in B$ and $j \in [m]$. All these edges (colored black in the figure) have length $\varepsilon < 1$.

The construction is summarized in Figures 3a, 3b.



(a) The general structure of the graph



(b) A closer look at an edge $(u, v) \in E$

Figure 3

In the resulting construction, the number of vertices, denoted by N, is $N = AK + Bn + ABm + 3$, and the number of edges, denoted by M, is at least $M \geq AB(Km + m) + AK + ABm + Bn$. Let OPT denote the cost of an optimal HL_∞ solution H for this instance. Then, by a standard pigeonhole principle argument, and since every edge is a unique shortest path, we get that $OPT \geq \frac{M}{N}$. We now set the parameters, as follows: $A = B = K = n^2$. With these values, we have $N = \Theta(n^4 \cdot m)$, $M = \Omega(n^6 \cdot m)$ and $OPT = \Omega(n^2)$.

We will describe an intended feasible solution for this instance, that will give an upper bound on OPT.

Let $I \subseteq [m]$ denote an optimal Set Cover of our original Set Cover instance, and let $I_j \in I$ denote the index of a chosen set that covers x_j. The HL solution is the following:

- $H_{r_{u,i}} = \{r_{u,i}\} \cup \left(\bigcup_{v \in B}\{S_{uv,j} : j \in I\}\right) \cup \{W_A, W_B, W_S\}$, for $u \in A$ and $i \in [K]$.
- $H_{x_{v,j}} = \{x_{v,j}\} \cup \left(\bigcup_{u \in A}\{S_{uv,I_j}\}\right) \cup \{W_A, W_B, W_S\}$, for $v \in B$ and $j \in [n]$.
- $H_{S_{uv,t}} = \{S_{uv,t}\} \cup \{r_{u,1}, ..., r_{u,K}\} \cup \{x_{v,1}, ..., x_{v,n}\} \cup \{W_A, W_B, W_S\}$, for $u \in A$, $v \in B$ and $t \in [m]$.
- $H_{W_q} = \{W_A, W_B, W_S\}$, for $q \in \{A, B, S\}$.

We now compute the sizes of these hub sets. We have:

- $|H_{r_{u,i}}| = B|I| + 4 = \Theta(n^2 \cdot |I|)$, for $u \in A$ and $i \in [K]$.
- $|H_{x_{v,j}}| = A + 4 = \Theta(n^2)$, for $v \in B$ and $j \in [n]$.
- $|H_{S_{uv,t}}| = K + n + 4 = \Theta(n^2)$, for $u \in A$, $v \in B$ and $t \in [m]$.
- $|H_{W_q}| = 3$, for $q \in \{A, B, S\}$.

Thus, we get that the value of the above solution is $\|H\|_\infty = Val = \Theta(n^2 \cdot |I|)$. We now show that the above is indeed a feasible solution. For that, we consider all possible pairs of vertices:

- $r_{u,i}$ - $r_{v,j}$: W_A is a hub for both vertices.
- $r_{u,i}$ - $S_{uv,j}$: $r_{u,i} \in H_{S_{uv,j}}$.
- $r_{u,i}$ - $S_{wv,j}$, $w \neq u$, $v \neq u$: W_S is a hub for both vertices.
- $r_{u,i}$ - $x_{v,j}$: S_{uv,I_j} is a hub for both vertices.
- $r_{u,i}$ - W_q, for $q \in \{A, B, S\}$: $W_q \in H_{r_{u,i}}$.
- $S_{uv,i}$ - $S_{u'v',j}$: W_S is a hub for both vertices.
- $S_{uv,i}$ - $x_{v,j}$: $x_{v,j} \in H_{S_{uv,i}}$.
- $S_{uv,i}$ - $x_{v',j}$, $u \neq v'$, $v \neq v'$: W_B (or W_S) is a hub for both vertices.
- $S_{uv,i}$ - W_q, for $q \in \{A, B, S\}$: $W_q \in H_{S_{uv,i}}$.
- $x_{v,i}$ - $x_{v',j}$: W_B is hub for both vertices.
- $x_{v,j}$ - W_q, for $q \in \{A, B, S\}$: $W_q \in H_{x_{v,j}}$
- W_q - $W_{q'}$, for $q, q' \in \{A, B, S\}$: $W_{q'} \in H_{W_q}$.

Thus, the proposed solution is indeed a feasible solution. Assuming now that we have an $f(n)$-approximation algorithm for HL_∞, we can obtain a solution H' of cost $\|H'\|_\infty \leq f(N) \cdot OPT \leq c \cdot f(N) \cdot n^2 \cdot |I|$. We will now show that we can extract a feasible solution for the original Set Cover instance, of cost $O(f(N)) \cdot |I|$. Let H' be the solution that the algorithm returns. As a reminder, we have already proved that $\|H'\|_\infty = \Omega(n^2)$. We first transform H' to a solution H'' that will look more like our intended solution, as follows:

- $H''_{S_{uv,t}} := H'_{S_{uv,t}} \cup \{r_{u,1}, ..., r_{u,K}\} \cup \{x_{v,1}, ..., x_{v,n}\} \cup \{W_A, W_B, W_S\}$, for $u \in A$, $v \in B$ and $t \in [m]$. We have $|H''_{S_{uv,t}}| \leq |H'_{S_{uv,t}}| + K + n + 3 \leq \|H'\|_\infty + n^2 + n + 3 = O(\|H'\|_\infty)$.
- $H''_{W_q} := H'_{W_q} \cup \{W_A, W_B, W_S\}$, for $q \in \{A, B, S\}$. We have $|H''_{W_q}| \leq |H'_{W_q}| + 3 = O(|H'_{W_q}|) = O(\|H'\|_\infty)$.
- We now look at $H'_{r_{u,i}}$. For every $x_j \in \mathcal{X}$, we pick (arbitrarily) a set $S(x_j) \in \mathcal{S}$ with $x_j \in S(x_j)$, that we will use to cover it. Now, if $x_{v,j} \in H'_{r_{u,i}}$, we remove $x_{v,j}$ from $H'_{r_{u,i}}$ and add $S_{uv}(x_j)$ to $H'_{r_{u,i}}$. This does not change the size of $H'_{r_{u,i}}$. We also add $S_{uv}(x_j)$ to the hub set of $x_{v,j}$. This increases the size of $H'_{x_{v,j}}$ by 1. The crucial observation here is that since we have decided in advance which set we will use to cover x_j, then $|H'_{x_{v,j}}|$ can only increase by 1, for every edge (u, v). Thus, the total increase in $|H'_{x_{v,j}}|$ is at most A, i.e. $|H''_{x_{v,j}}| \leq |H'_{x_{v,j}}| + n^2 = O(\|H'\|_\infty)$.

The above transformed solution, as shown, has the same (up to constant factors) cost as the solution that the algorithm returns, i.e. $\|H''\|_\infty = O(\|H'\|_\infty) = O(f(N)) \cdot n^2 \cdot |I|$, and is clearly feasible.

In order to recover a good Set Cover solution, we look at the sets $H''_{r_{u,i}} \cap \mathcal{S}_{uv}$. Each such intersection can be viewed as a subset $C_{u,v,i}$ of $\mathcal{S}$. Let $Z_{u,v,i}$ denote the number of elements that are not covered by $C_{u,v,i}$, i.e. $Z_{u,v,i} = |\mathcal{X} \setminus (\bigcup_{S \in C_{u,v,i}} S)|$. Our goal is to show that there exists a $\{u, v, i\}$ such that $|C_{u,v,i}| + Z_{u,v,i} = O(\|H''\|_\infty / n^2)$. Since there is a polynomial number of choices of $\{u, v, i\}$, we can then enumerate over all choices and find a Set Cover with cost $O(f(N)) \cdot |I|$.

To prove that such a good choice exists, we will make a uniformly random choice over $\{u, v, i\}$, and look at the expected value $\mathbb{E}\left[|C_{u,v,i}| + Z_{u,v,i}\right]$. We have $\mathbb{E}\left[|C_{u,v,i}| + Z_{u,v,i}\right] = \mathbb{E}\left[|C_{u,v,i}|\right] + \mathbb{E}[Z_{u,v,i}]$. We look separately at the two terms. We make the following 2 observations:

$$\sum_{v \in B} |C_{u,v,i}| \leq |H''_{r_{u,i}}| = O(\|H'\|_\infty),$$

and

$$\sum_{u \in A} \sum_{i \in K} Z_{u,v,i} \leq \sum_{j \in [n]} |H''_{x_{v,j}}| = n \cdot O(\|H'\|_\infty).$$

The second observation follows from the fact that for any given $r_{u,i}$ and edge (u, v), the uncovered elements $x_{v,j}$ must have $r_{u,i} \in H''_{x_{v,j}}$. With these, we have

$$\begin{aligned}\mathbb{E}[|C_{u,v,i}|] &= \frac{1}{ABK} \sum_{u \in A, i \in [K]} \sum_{v \in B} |C_{u,v,i}| \\ &\leq \frac{1}{ABK} \cdot AK \cdot O(\|H'\|_\infty) = O(\|H'\|_\infty / B).\end{aligned}$$

Similarly,

$$\begin{aligned}\mathbb{E}[Z_{u,v,i}] &= \frac{1}{ABK}\sum_{v\in B}\sum_{u\in A}\sum_{i\in K} Z_{u,v,i} \\ &\leq \frac{1}{ABK}\cdot B\cdot n\cdot O(\|H'\|_\infty) \\ &= \frac{n}{AK}\cdot O(\|H'\|_\infty).\end{aligned}$$

Thus, we get that $\mathbb{E}\left[|C_{u,v,i}| + Z_{u,v,i}\right] = \left(\frac{1}{B} + \frac{n}{AK}\right)\cdot O(\|H'\|_\infty) = O(\|H'\|_\infty/n^2) = O(f(N))\cdot|I|$. This means that there exists a choice of $\{u,v,i\}$ such that the corresponding Set Cover has size $O(f(N))\cdot|I|$. As already mentioned, there are polynomially many choices, so we can enumerate them and find the appropriate $\{u,v,i\}$, and, thus, recover a Set Cover solution for our original Set Cover instance of cost $O(f(N))\cdot|I|$, where, as already stated, $N = \Theta(n^4\cdot m)$. □

COROLLARY 6.2. *It is NP-hard to approximate HL_∞ to within a factor better than $\Omega(\log n)$.*

Proof. The previous theorem gives an $O(f(O(n^4m))$-approximation algorithm for Set Cover, given that an $f(n)$-approximation algorithm for HL_∞ exists. If we assume that there exists such an algorithm with $f(n) = o(\log n)$, then we could use it to approximate Set Cover within a factor $o(\log O(n^4m)) = o(\log \mathtt{poly}(n)) = o(\log n)$, and this is impossible, assuming that P $\neq$ NP. □

Acknowledgments

We would like to thank Robert Krauthgamer and Konstantin Makarychev for useful discussions, and the anonymous referees for their valuable comments. Research was supported by the second author's NSF grants CAREER CCF-1150062 and IIS-1302662.

References

[1] Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato Fonseca F. Werneck. VC-Dimension and Shortest Path Algorithms. In *Proc. of the International Colloquium on Automata, Languages, and Programming*, pages 690–699, 2011.

[2] Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato Fonseca F. Werneck. HLDB: location-based services in databases. In *Proc. of the International Conference on Advances in Geographic Information Systems*, pages 339–348, 2012.

[3] Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. Alternative routes in road networks. *ACM Journal of Experimental Algorithmics*, 18, 2013.

[4] Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato Fonseca F. Werneck. A Hub-Based Labeling Algorithm for Shortest Paths in Road Networks. In *Proc. of the International Symposium on Experimental Algorithms*, pages 230–241, 2011.

[5] Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato Fonseca F. Werneck. Hierarchical Hub Labelings for Shortest Paths. In *Proc. of the European Symposium on Algorithms*, pages 24–35, 2012.

[6] Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato Fonseca F. Werneck. Highway Dimension, Shortest Paths, and Provably Efficient Algorithms. In *Proc. of the Symposium on Discrete Algorithms*, pages 782–793, 2010.

[7] Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms*, 2(2):153–177, 2006.

[8] Maxim A. Babenko, Andrew V. Goldberg, Anupam Gupta, and Viswanath Nagarajan. Algorithms for Hub Label Optimization. In *Proc. of the International Colloquium on Automata, Languages, and Programming*, pages 69–80, 2013.

[9] Maxim A. Babenko, Andrew V. Goldberg, Haim Kaplan, Ruslan Savchenko, and Mathias Weller. On the Complexity of Hub Labeling. In *Proc. of the International Symposium on Mathematical Foundations of Computer Science*, pages 62–74, 2015.

[10] Hannah Bast, Daniel Delling, Andrew V. Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. Route planning in transportation networks. *CoRR*, abs/1504.05140, 2015.

[11] Holger Bast, Stefan Funke, and Domagoj Matijevic. TRANSIT - ultrafast shortest-path queries with linear-time preprocessing. In *9th DIMACS Implementation Challenge*, 2006.

[12] Daniel Berend and Tamir Tassa. Improved bounds on bell numbers and on moments of sums of random variables. *Probability and Math. Statistics*, 30:185–205, 2010.

[13] Melvin A. Breuer. Coding the vertexes of a graph. *IEEE Trans. Information Theory*, 12(2):148–153, 1966.

[14] Melvin A Breuer and Jon Folkman. An unexpected result in coding the vertices of a graph. *Journal of Mathematical Analysis and Applications*, 20(3):583–600, 1967.

[15] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and Distance Queries via 2-Hop Labels. *SIAM J. Comput.*, 32(5):1338–1355, 2003.

[16] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proc. of the Symposium on Theory of Computing*, pages 624–633, 2014.

[17] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[18] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *J. Algorithms*, 53(1):85–112, 2004.

[19] P.C. Gilmore. Families of sets with faithful graph

representations. *Res. Note N. C. (2nd ed.)*, 184, 1962.
[20] R. L. Graham and H. O. Pollak. On embedding graphs in squashed cubes. *Lecture Notes in Mathematics*, 303:99–110, 1972.
[21] András Gyárfás and Jenö Lehel. A Helly-type problem in trees. *Colloq. Math. Soc. J. Bolyai.*, 4, 1970.
[22] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit Representation of Graphs. *SIAM J. Discrete Math.*, 5(4):596–603, 1992.
[23] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.
[24] David Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33(3):167–176, 2000.
[25] Ran Raz and Shmuel Safra. A Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. In *Proc. of the Symposium on Theory of Computing*, pages 475–484, 1997.
[26] Colin White. Lower bounds in the preprocessing and query phases of routing algorithms. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 1013–1024, 2015.
[27] Peter M. Winkler. Proof of the squashed cube conjecture. *Combinatorica*, 3(1):135–139, 1983.

A $\mathbf{HL}_p$ on graphs with unique shortest paths

In this section, we analyze Algorithm 4.1, assuming that the objective function is $\left(\sum_{u\in V}|H_u|^p\right)^{1/p}$, for arbitrary fixed $p \geq 1$. To do so, we first modify $\mathbf{LP_1}$ and turn it into a convex program, with the same constraints and the following objective function:

$$\min: \left(\sum_{u\in V}\left(\sum_{v\in V} x_{uv}\right)^p\right)^{1/p}.$$

To analyze the performance of Algorithm 4.1 in this case, we need the following theorem by Berend and Tassa [12].

THEOREM A.1. (THEOREM 2.4, [12]) *Let $X_1, ..., X_t$ be a sequence of independent random variables for which $\mathbf{Pr}[0 \leq X_i \leq 1] = 1$, and let $X = \sum_{i=1}^t X_i$. Then, for all $p \geq 1$,*

$$(\mathbb{E}[X^p])^{1/p} \leq 0.792 \cdot v(p) \cdot \frac{p}{\ln(p+1)} \cdot \max\{\mathbb{E}[X]^{1/p}, \mathbb{E}[X]\},$$

where

$$v(p) = \left(1 + \frac{1}{\lfloor p \rfloor}\right)^{\frac{\{p\}\cdot(1-\{p\})}{p}},$$

with $\{p\}$ denoting the fractional part of p.

THEOREM A.2. *For any $p \geq 1$, Algorithm 4.1 is an $O\left(\frac{p}{\ln(p+1)} \cdot \log D\right)$-approximation algorithm for HL_p.*

Proof. In order to simplify the analysis and be able to use the above theorem (Theorem A.1) as is, we slightly modify Algorithm 4.1 as follows. After we obtain the set of pre-hubs $\{\widehat{H}_u\}_{u\in V}$ at step 2, we consider the rooted tree T_u, defined as $T_u = \bigcup_{u'\in\widehat{H}_u} P_{uu'}$ (observe that this is indeed a tree), for every $u \in V$, and define $F_u \subset V(T_u)$ to be the set of vertices of T_u whose degree (in T_u) is at least 3. The modified algorithm then adds the additional step (2′): "$\widehat{H}'_u := \widehat{H}_u \cup F_u$", and then continues the execution of Algorithm 4.1 at step 3, as usual, using the modified sets $\widehat{H}'_u$. Let ALG be the cost of original algorithm, and ALG' be the cost of this modified algorithm. It is not hard to prove that it always holds $ALG \leq ALG'$. It is also easy to see that, since all leaves of T_u are pre-hubs of the set $\widehat{H}_u$, we must have $|F_u| \leq |\widehat{H}_u|$, and so $|\widehat{H}'_u| \leq 2 \cdot |\widehat{H}_u|$.

Let $\mathcal{P}_u$ be the collection of subpaths of T_u defined as follows: P belongs to $\mathcal{P}_u$ if P is a path between consecutive pre-hubs u'' and u' of $\widehat{H}'_u$, with u'' being an ancestor of u' in T_u, and no other pre-hub $u''' \in \widehat{H}_u$ appears in P. For convenience, we exclude the endpoint u'' that is closer to u: $P = P_{u''u'} - u''$. Note that any such path P is uniquely defined by the pre-hub u' of u, and so we will write $P = P_{(uu')}$. The modification we made in the algorithm allows us now to observe that $P \cap P' = \varnothing$, for $P, P' \in \mathcal{P}_u$, $P \neq P'$.

Let ALG' be the cost of the solution $\{H_u\}_u$ that the modified algorithm returns. We have

$$\mathbb{E}[ALG'] = \mathbb{E}\left[\left(\sum_{u\in V}|H_u|^p\right)^{1/p}\right] \leq \left(\sum \mathbb{E}[|H_u|^p]\right)^{1/p},$$

where we utilize Jensen's inequality. We can write $|H_u| \leq \sum_{v\in\widehat{H}'_u} X^u_v$, where X^u_v is the random variable indicating how many vertices are added to H_u "because of" the pre-hub v. Observe that we can write X^u_v as follows: $X^u_v = \sum_{w\in P_{(uv)}} Y^{uv}_w$, with Y^{uv}_w being 1 if w is added in H_u, and zero otherwise. The modification that we made in the algorithm implies, as already observed, that any variable Y^{uv}_w, $w \in P_{(uv)}$, is independent from $Y^{uv'}_{w'}$, $w' \in P_{(uv')}$, for $v \neq v'$, as the corresponding paths $P_{(uv)}$ and $P_{(uv')}$ are disjoint.

Let $\pi_{uv} : [|P_{(uv)}|] \to P_{(uv)}$ be the permutation we obtain when we restrict π to the vertices of $P_{(uv)}$. We can then write $\sum_{w\in P_{(uv)}} Y^{uv}_w = \sum_{i=1}^l Z^{uv}_i$, $l = |P_{(uv)}|$, where Z^{uv}_i is 1 if the i-th vertex considered by the algorithm that belongs to $P_{(uv)}$ (i.e. the i-th vertex of permutation π_{uv}) is added to H_u and zero otherwise. It is easy to see that $\Pr[Z^{uv}_i = 1] = 1/i$. We now need one last observation. We have $\Pr[Z^{uv}_i = 1 \mid Z^{uv}_1, ..., Z^{uv}_{i-1}] = 1/i$. To see this, note that the variables Z^{uv}_i do not reveal which particular vertex is picked from the

permutation at each step, but only the relative order of the current draw (i.e. i-th random choice) with respect to the current best draw (where best here means the closest vertex to v that we have seen so far, i.e. in positions $\pi_{uv}(1), ..., \pi_{uv}(i-1)$). Thus, regardless of the relative order of $\pi_{uv}(1), ..., \pi_{uv}(i-1)$, there are exactly i possibilities to extend that order when the permutation picks $\pi_{uv}(i)$, each with probability $1/i$. This shows that the variables $\{Z_i^{uv}\}_i$ are independent, and thus all variables $\{Z_i^{uv}\}_{v \in \widehat{H}'_v,\, i \in [|P_{(uv)}|]}$ are independent.

We can now apply Theorem A.1. This gives

$$\mathbb{E}[|H_u|^p] \leq \mathbb{E}\left[\left(\sum_{v \in \widehat{H}'_u} \sum_{i=1}^{|P_{(uv)}|} Z_i^{uv}\right)^p\right] \leq \left(0.792 \cdot v(p) \cdot \frac{p}{\ln(p+1)}\right)^p \cdot \mathrm{Harm}_D^p \cdot |\widehat{H}'_u|^p.$$

Here, $\mathrm{Harm}_D = \sum_{i=1}^{D} \frac{1}{i} = \log D + O(1)$ is the D-th harmonic number. Thus,

$$\mathbb{E}[ALG'] \leq 0.792 \cdot \frac{v(p) \cdot p}{\ln(p+1)} \cdot \mathrm{Harm}_D \cdot \left(\sum_{u \in V} |\widehat{H}'_u|^p\right)^{1/p}$$

$$\leq \frac{0.792 \cdot v(p) \cdot p}{\ln(p+1)} \cdot \mathrm{Harm}_D \cdot \left(\sum_{u \in V} 4^p \cdot \left(\sum_{v \in V} x_{uv}\right)^p\right)^{1/p}$$

$$\leq c \cdot \frac{p}{\ln(p+1)} \cdot \mathrm{Harm}_D \cdot OPT_{\mathrm{REL}},$$

where OPT_{REL} is the optimal value of the convex relaxation, and $c \leq 7$ is some constant. □

The algorithm can be derandomized using the method of conditional expectations (see Remark 4.1).

B Any "natural" rounding scheme cannot break the $O(\log n)$ barrier for HL_1 on graphs with unique shortest paths and diameter D

In this section, we show that any rounding scheme that may assign $v \in H_u$ only if $x_{uv} > 0$ gives $\Omega(\log n)$ approximation, even on graphs with shortest-path diameter $D = O(\log n)$. For that, consider the following tree T, which consists of a path $P = \{1, ..., k\}$ of length $k = 3t$, $t \in \mathbb{N} \setminus \{0\}$, and two stars $\mathcal{A}$ and $\mathcal{B}$, with $N = \binom{k}{2t}$ leaves each (each leaf corresponding to a subset of $[k]$ of size exactly $2t$). The center a of $\mathcal{A}$ is connected to vertex "1" of P and the center b of $\mathcal{B}$ is connected to vertex "k" of P. The total number of vertices of T is $n = 2N + 2 + k$, which implies that $t = \Omega(\log n)$.

Consider now the following LP solution (all variables not assigned below are set to zero):

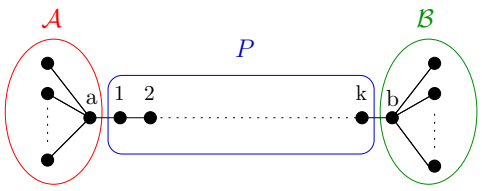


Figure 4: An instance that cannot be rounded well with any "natural" rounding scheme

- $x_{uu} = 1$, for all $u \in T$.
- $x_{Sa} = 1$, for all $S \in \mathcal{A}$.
- $x_{Wb} = 1$, for all $W \in \mathcal{B}$.
- $x_{Si} = 1/t$, for all $S \in \mathcal{A}$, $i \in S \subseteq P$.
- $x_{Wi} = 1/t$, for all $W \in \mathcal{B}$, $i \in W \subseteq P$.
- $x_{ab} = x_{ba} = 1$.
- $x_{ia} = x_{ib} = 1$, for all $i \in [k]$.
- $\{x_{ij}\}_{i,j \in [k]}$ is an optimal solution for P.

Observe that the above solution is indeed a feasible fractional solution. Its cost is at most $n + 3(|\mathcal{A}| + |\mathcal{B}|) + 2 + 2k + c \cdot k \cdot \log k = \Theta(n)$, for some constant c. Suppose now that we are looking for a rounding scheme that assigns $v \in H_u$ only if $x_{uv} > 0$, and let's assume that there exists a vertex $S \in \mathcal{A}$ whose resulting hub set satisfies $|H_S \cap P| < t$. We must also have $H_S \cap \mathcal{B} = \varnothing$, since $x_{Su} = 0$ for all $u \in \mathcal{B}$. This implies that there exists a $W \in \mathcal{B}$ such that $W \cap H_S = \varnothing$. Since the above fractional solution assigns non-zero values only to x_{Wi} with $i \in W$ and x_{Wb}, this means that $x_{Wi} = 0$ for all $i \in H_S$. Thus, the resulting hub set cannot be feasible, which implies that any rounding that satisfies the aforementioned property and returns a feasible solution must satisfy $|H_S \cap P| \geq t$ for all $S \in \mathcal{A}$ (similarly, the same holds for all $W \in \mathcal{B}$). This means that the returned solution has cost $\Omega(n \cdot t) = \Omega(n \cdot \log n)$, and so the approximation factor must be at least $\Omega(\log n)$.