Hardness of Approximation for Orienteering with Multiple Time Windows

Naveen Garg* Sanjeev Khanna† Amit Kumar‡

Abstract

Vehicle routing problems are a broad class of combinatorial optimization problems that can be formulated as the problem of finding a tour in a weighted graph that optimizes some function of the visited vertices. For instance, a canonical and extensively studied vehicle routing problem is the *orienteering* problem where the goal is to find a tour that maximizes the number of vertices visited by a given deadline. In this paper, we consider the computational tractability of a well-known generalization of the orienteering problem called the Orient-MTW problem. The input to Orient-MTW consists of a weighted graph $G(V, E)$ where for each vertex $v \in V$ we are given a set of time instants $T_v \subseteq [T]$, and a source vertex s. A tour starting at s is said to visit a vertex v if it transits through v at any time in the set T_v. The goal is to find a tour starting at the source vertex that maximizes the number of vertices visited. It is known that this problem admits a quasi-polynomial time $O(\log \mathsf{OPT})$-approximation ratio where OPT is the optimal solution value but until now no hardness better than an APX-hardness was known for this problem.

Our main result is an $\Omega\left(\frac{\log\log n}{\log\log\log n}\right)$-hardness for this problem that holds even when the underlying graph G is an undirected tree. This is the first super-constant hardness result for the Orient-MTW problem. The starting point for our result is the hardness of the SetCover problem which is known to hold on instances with a special structure. We exploit this special structure of the hard SetCover instances to first obtain a new proof of the APX-hardness result for Orient-MTW that holds even on trees of depth 2. We then recursively amplify this constant factor hardness to an $\Omega\left(\frac{\log\log n}{\log\log\log n}\right)$-hardness, while keeping the resulting topology to be a tree. Our amplified hardness proof crucially utilizes a delicate concavity property which shows that in our encoding of SetCover instances as instances of the Orient-MTW problem, whenever the optimal cost for SetCover instance is large, any tour, no matter how it allocates its time across different sub-trees, can not visit too many vertices overall. We believe that this reduction template may also prove useful in showing hardness of other vehicle routing problems.

*Department of Computer Science & Engineering, IIT Delhi, email: naveen@cse.iitd.ac.in

†Department of Computer and Information Science, University of Pennsylvania, email:sanjeev@cis.upenn.edu

‡Department of Computer Science & Engineering, IIT Delhi, email: amitk@cse.iitd.ac.in

1 Introduction

Vehicle routing problems, which seek to find a set of optimal routes for a set of vehicles in order to service demands, have been widely studied in the operations research community (see e.g. [16]). One of the most well-known special cases of this problem is the traveling salesman problem (TSP) [1]. In TSP and related problems, the goal is to design minimum cost routes which visit all the demands. In the orienteering problem, we are given a metric space (or an edge weighted graph) and a budget B. The goal is to find a tour of length at most B and maximize the number (or total weight) of vertices which are visited by it. The first constant factor approximation for the orienteering problem was given by Blum et al. [3]. Since this result, several generalizations of the orienteering problem have been studied. In orienteering with multiple time windows (Orient-MTW), each vertex v specifies a subset T_v of available time slots when the tour can visit it. The goal is again to find a tour which maximizes the number of visited nodes. The orienteering problem is a special case of Orient-MTW where all time windows T_v are of the form $[0, B]$. No constant factor approximation algorithms are known for Orient-MTW even when the allowable set of timeslots for each vertex is of the form $[0, B_v]$, for some parameter B_v, and the underlying metric is a tree metric. Given that a constant factor approximation has remained elusive even for this special case of Orient-MTW, one might be tempted to believe that the general case of Orient-MTW may provably exhibit a much stronger hardness. However, Chekuri and Pal [6] showed that there is an $O(\log \mathsf{OPT})$-approximation algorithm for Orient-MTW that runs in quasi-polynomial time, where OPT denotes the number of vertices visited by an optimal tour.

While a logarithmic approximation ratio is achiev-

able for Orient-MTW, the known hardness results do not rule out the possibility of a constant factor approximation algorithm for this problem. In this paper, we give the first super-constant hardness result for Orient-MTW. Specifically, we show that under the standard complexity theoretic assumption, namely $NP \not\subseteq DTIME(n^{O(\log n)})$, any polynomial time algorithm for Orient-MTW must have $\Omega(\log \log n)$-approximation ratio, even when the underlying metric is a *tree*. We establish this hardness result by a reduction from the set cover problem (SetCover). As shown by Feige [9], it is hard to distinguish between the following instances of SetCover: (i) there are K disjoint sets which cover the underlying universe of elements, or (ii) any collection of cK sets, where $c > 1$ is a parameter in a suitable range, covers at most a $1-\frac{1}{e^c}$-fraction of elements. We use such instances of SetCover to create instances of Orient-MTW where in the first case, there is a tour which can visit all the vertices, whereas in the second case, any tour must leave out a large fraction of vertices. The challenge in the analysis comes from the fact that the tour may try to achieve a trade-off by spending more time on some sub-trees (and less on some others) – in fact such strategies are often used in designing approximation algorithms for maximization problems. We show that the instances created by our reduction have certain concavity properties, and therefore, investing additional time in a sub-tree does not pay off sufficiently. Note that we cannot study such concavity properties for a *fractional* tour (which can be thought of as convex combination of integral tours), because fractional SetCover can be solved in polynomial time, and so one cannot get any hardness out of this reduction. Therefore, the use of the concavity property is more subtle – we use an auxiliary (continuous) concave function, and show that any tour corresponds to integral assignment of parameters to this concave function. We feel that the novel techniques used in the reduction may be useful for showing hardness for special cases of Orient-MTW as well.

1.1 Related Work The Orienteering problem is known to be APX-hard [3]. Blum et al. [3] gave the first constant factor approximation algorithm for this problem with approximation ratio of 4. Bansal et al. [2] gave an improved algorithm with approximation ratio of 3, which was further improved to $(2+\varepsilon)$ for any positive constant ε by Chekuri et al. [4]. Friggstad et al. [10] gave an LP rounding based 3-approximation algorithm for Orienteering. A PTAS is known for this problem when the points lie in a constant dimensional Euclidean metric [7]. Recall that in the more general Orient-MTW problem, T_v denotes the set of allowable timeslots for a vertex v. In the special case when T_v is of the form $[0, B_v]$ for each vertex v (the variable deadline case), Bansal et al. [2] gave an $O(\log \mathsf{OPT})$-approximation algorithm. Chekuri and Kumar [5] gave a constant factor approximation algorithm for the special case of this problem when the set of deadlines B_v has constant cardinality.

They also gave an $O(\log^2 \mathsf{OPT})$ algorithm when the set T_v is given by a single time interval (single time-window case). Chekuri et al. [4] gave $O(\max(\log \mathsf{OPT}, \log L))$-approximation where L is the ratio of the maximum to the minimum length of the time window for any vertex. For the case when each of the sets T_v consists of at most k time intervals, the only known approximation algorithm incurs a multiplicative factor of k over the corresponding algorithm for the single time-window case.

The extension to directed graphs is less well-understood. Nagarajan and Ravi [13] gave an $O(\alpha \cdot \log n)$-approximation algorithms for Orienteering in directed graphs, where α is the integrality gap for the Held-Karp LP relaxation for the Asymmetric Traveling Salesman Problem. Svensson et al. [15] showed that α is $O(1)$, which implies an $O(\log n)$-approximation algorithm for Orienteering in directed graphs. Nagarajan and Ravi [13] also gave $O(\log^3 n)$-approximation algorithm for the special case of Orient-MTW where each vertex specifies a single time window. Similar results for these problems were obtained by Chekuri et al. [4]. They gave an $O(\log^2 \mathsf{OPT})$-approximation for orienteering in directed graphs. They also gave $O(\log^3 \mathsf{OPT})$ and $O(\log^4 \mathsf{OPT})$-approximation algorithms for the variable deadline and single time-window special cases of Orient-MTW in directed graphs.

The recursive greedy approach of Chekuri and Pal [6] gives a quasi-polynomial time $O(\log \mathsf{OPT})$-approximation algorithm for Orient-MTW, even in the case of directed graphs.

1.2 Our Result and Techniques Our main result is the following.

THEOREM 1.1. *Unless NP is contained in $DTIME(n^{O(\log n)})$, there is no polynomial-time $o\left(\frac{\log \log n}{\log \log \log n}\right)$-approximation algorithm for* Orient-MTW *even on trees.*

This is the first super-constant hardness of approximation result for Orient-MTW in undirected graphs. Our starting point is establishing an APX-hardness for Orient-MTW on trees of depth 2. Even though the APX-hardness of Orient-MTW in general graphs follows from the known APX-hardness of the Orienteering problem [3], the very restrictive nature of the tree metric

space used in our reduction plays a crucial role in obtaining the final hardness result of $\Omega\left(\frac{\log\log n}{\log\log\log n}\right)$-hardness result on trees of higher depth.

We describe here some intuition about the APX-hardness reduction for Orient-MTW on trees, and highlight some technical difficulties that already arise in showing this weaker hardness. We reduce from an instance $\mathcal{I}$ of the SetCover problem, consisting of a ground set U and a collection $S_1, \ldots, S_m$ of subsets of U. The first idea is to construct an instance $\mathcal{I}'$ of Orient-MTW as follows: $\mathcal{I}'$ is just a star graph with a root r and each leaf is identified with a unique element of U. Further if a leaf v belongs to the sets $S_{i_1}, \ldots . S_{i_s}$, we define T_v, the timeslots when v can be visited, to be $\{i_1, \ldots, i_s\}$. Further, we set the length of every (root-leaf) edge to 0. The timeline is $[1, m]$. Clearly, if there is a set cover of size K, say $S_{i_1}, \ldots, S_{i_K}$, then there is a tour which visits all the leaf vertices during one of the times in $\{i_1, \ldots, i_K\}$, i.e., it needs to spend only K timeslots in $[1, m]$ to visit all the leaf vertices. Now, to ensure that any tour spends only K distinct timeslots in this star graph, we need to add more copies of this star graph (as many as m/K) so that a tour can spend only K timeslots in each such copy of the star graph. Any two such copies will be at distance 1 from each other such that a tour can be in only one of these copies at any integer time instant. However, there are several issues that arise. First, consider two leaf vertices v_1 and v_2 belonging to two different copies of the star graph, but they correspond to the same element in U. Now, if the tour can cover v_1 at time t, it cannot also cover v_2 at the same time. In order to get around this, we relabel the indices of the sets for each star graph so that this issue does not arise – we use the structure of the hard instances of the SetCover problem [9], and it turns out that this relabeling operation is simply a cyclic shift operation. Second, now that we have multiple copies of the star graph, a tour may choose to spend varying amount of time in each such copy (i.e., it need not spend K integer time instants in each such copy). For this, we show that we can use the properties of the hard SetCover instances to show that any such trade-off can only hurt the performance of the tour.

In going from the above APX-hardness result to the $\Omega(\log\log n)$ hardness, we create a sequence of trees of increasing depth in an iterative manner where each tree in the sequence is obtained from its predecessor by replacing the leaf nodes with a suitably modified version of the two-level hard instance described above. If we denote by $H^{(l)}$ the tree obtained after l iterations, then our construction ensures that in a YES instance, a visit to a leaf vertex in the tree $H^{(l-1)}$ in this sequence can be replaced with a tour of the two-level tree in the tree $H^{(l)}$. On the other hand, in a NO instance, we show that no matter how a tour allocates time across different subtrees, it can visit only a $O(1/\log l)$-fraction of the leaves. The analysis of the NO instances created above is the heart of our technical contribution and relies on a delicate concavity property which essentially shows that any tour that spends more time in some subtrees at the expense of others, can not do much better than the tours which spend time uniformly across various subtrees.

1.3 Organization We formally define the Orient-MTW problem in Section 2 as well as introduce the family of hard SetCover instances that serve as the starting point for our hardness results. In Section 3, we show that Orient-MTW is APX-hard even for depth-2 trees. In Section 4, we amplify this reduction to an $\Omega(\log\log n)$-hardness on higher depth trees, assuming a key technical result about the concavity of a recursively defined function. This concavity result helps us in proving that an algorithm cannot gain much by distributing effort non-uniformly in the tree. We prove this concavity result in Section B.

2 Preliminaries

2.1 The Orient-MTW Problem We are given an undirected graph $G = (V, E)$ with edge lengths ℓ_e and a positive integer T. Further, every vertex v specifies a subset T_v of $[T]$. We are also given a special source vertex s in V. A solution consists of a tour $\mathcal{T}$ starting from s which can be thought of as the path traversed by a unit speed vehicle. In order to allow the possibility of the vehicle just "idling" at any vertex, we define a tour formally as follows: it is a sequence $s = v_1, t_1, v_2, t_2, \ldots, v_k, t_k$, where each v_i is a vertex and each t_i is a positive real number. Further for each $i = 2, \ldots, k$, $(v_{i-1}, v_i) \in E$. Note that a vertex could be repeated any number of times in this sequence. The interpretation of this sequence is as follows: the vehicle starts at v_1 at time 0, waits there for t_1 amount of time, and then travels to v_2 at unit speed (along the edge (v_1, v_2)), waits there for t_2 time units and so on. Thus, the vehicle will reach the vertex v_i in this sequence at $\sum_{j=1}^{i-1} t_j + \sum_{j=1}^{i-1} \ell_{e_j}$ time, where e_j denotes the edge (v_j, v_{j+1}). The tour is said to *visit* a vertex v if it is present at v at any of the time instances in T_v (note that the tour can be at a vertex v at multiple time instances in T_v, but v gets counted only once). The goal is to find a tour which maximizes the number of visited vertices.

2.2 Hard Instances of the SetCover Problem Recall that an instance of the SetCover problem can be described by a pair $(U, \mathcal{S})$, where U is the ground set and $\mathcal{S}$ is a collection of subsets of U. The goal is to pick a mini-

mum cardinality collection of sets in $\mathcal{S}$ whose union is U. Our hardness results will utilize the following properties of hard instances of the SetCover problem as shown by Lund and Yannakakis [12]. The proof is given in the appendix for sake of completeness. Although there have been improvements on the SetCover hardness [9, 8], we rely on specific properties of the reduction in [12].

THEOREM 2.1. *Given a 3-SAT formula ϕ of size n, there is a reduction algorithm that runs in $n^{O(\log\log n)}$ time, and outputs an instance $\mathcal{I}_\phi = \{(U, \mathcal{S}), K\}$ of* SetCover *of size $N = n^{O(\log\log n)}$ with following properties:*

(a) All sets in $\mathcal{S}$ have the same size.

(b) The sets in $\mathcal{S}$ can be partitioned into disjoint collections $\mathcal{G}_1, \ldots, \mathcal{G}_K$ of equal cardinality.

Moreover, the SetCover *instance created by the reduction above behaves as follows:*

- *(Completeness) If ϕ is satisfiable, then there exists disjoint sets $S_i \in G_i$, for $i = 1, \ldots, K$, such that $\cup_{i=1}^{K} S_i = U$.*
- *(Soundness) If ϕ is not satisfiable, then any collection of $c \cdot K$ sets in $\mathcal{S}$ will leave out at least $\frac{1}{e^{80c}}$ fraction of elements elements in U uncovered, where c is any positive real number lying in the range $(0, \frac{\log n}{C}]$ for some universal constant C.*

We will refer to a SetCover instance above as a *YES-instance* if it satisfies the completeness property and as a *NO-instance* otherwise.

3 APX Hardness of Orient-MTW on Trees

In this section we show that there is a positive constant $c > 1$ such that assuming $NP \not\subseteq DTIME(n^{O(\log\log n)})$, there is no polynomial time c-approximation algorithm for the Orient-MTW problem even when the metric space is given by a tree of depth 2.

We will give a reduction from the SetCover instances which satisfy the properties mentioned in Theorem 2.1 to instances of the Orient-MTW problem. Fix any SetCover instance $\mathcal{I} = \{(U, \mathcal{S}), K\}$ with the collections $\mathcal{G}_1, \ldots, \mathcal{G}_K$ as in the statement of Theorem 2.1. Let m denote $|\mathcal{S}|$, and so, each of the collections $\mathcal{G}_i$ has $g := m/K$ sets. We assign an index to each set in $\mathcal{S}$ such that the sets in the same collection $\mathcal{G}_i$ are assigned consecutive indices. In other words, let us denote the sets in $\mathcal{G}_i$ as $S_{(i-1)\cdot g+1}, \ldots, S_{i\cdot g}$. For ease of notation, let g_i denote $i \cdot g$. So the sets in $\mathcal{G}_i$ are $S_{g_{i-1}+1}, \ldots, S_{g_i}$.

We now create from $\mathcal{I}$ an instance $\mathcal{I}'$ of Orient-MTW where the graph G in $\mathcal{I}'$ will be a two level rooted tree. The root vertex r has g children, labelled $w^1, \ldots, w^g$. The length of each of these edges is 1/2. Each of the vertices w^i has n children, where n denotes $|U|$ – call these children $v_1^i, \ldots, v_n^i$. We shall denote the set U as $\{v_1, \ldots, v_n\}$, and so the vertex v_j^i will *correspond* to the element v_j. The length of the edge between w^i and v_j^i is 0 for all i, j. It remains to describe the subsets $T_{v_j^i}$ for all the leaf vertices in the tree (all other vertices do not require to be visited – we can define their T_v sets to be empty).

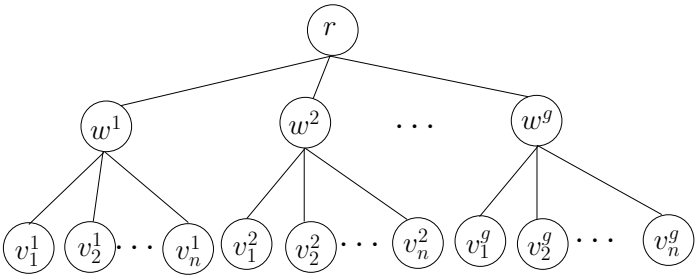

Figure 1: Reduction from SetCover to Orient-MTW: Suppose $g = 10$. Then $\mathcal{G}_1 = \{S_1, \ldots, S_{10}\}, \mathcal{G}_2 = \{S_{11}, \ldots, S_{20}\}$ and so on. If the element v_2 appears in the set S_5 in $\mathcal{G}_1$ and in the set S_{28} in $\mathcal{G}_3$, then $T_{v_2^1} = \{5, 28\}, T_{v_2^2} = \{6, 29\}, T_{v_2^3} = \{7, 30\}, T_{v_2^4} = \{8, 21\}, \ldots, T_{v_2^{10}} = \{4, 27\}$.

Before we describe these sets, we define an operation, which is similar to the mod operation, but *wraps* a number around a range.

DEFINITION 3.1. *Given a range $[a, b]$, where $a < b$ are non-negative integers, and a non-negative integer $x \geq a$, define x `mod` $[a, b]$ as $(x - a)$ `mod` $(b - a) + a$. Note that x `mod` $[a, b]$ always lies in the range $\{a, a+1, \ldots, b-1\}$.*

Consider the leaf vertex v_j^1 (which is the child of w^1 that corresponds to the element v_j in the SetCover instance). Let $S_{\ell_1}, \ldots, S_{\ell_r}$ be the sets in $\mathcal{S}$ containing v_j. Assume that these sets belong to the groups $\mathcal{G}_{s_1}, \ldots, \mathcal{G}_{s_r}$ respectively. We define $T_{v_j^1}$ as $\{\ell_1, \ldots, \ell_r\}$. The definition of the corresponding set for v_j^i, i.e. $T_{v_j^i}$, is more nuanced. Note that the indices $\ell_k \in [g_{s_k-1}+1, g_{s_k}]$ for each $k \in 1, \ldots, r$. We would like $T_{v_j^i}$ to be same as $T_{v_j^1}$ *shifted* by $(i-1)$ units, i.e., $\{\ell_1 + (i-1), \ldots, \ell_r + (i-1)\}$. However the problem with this definition is that $\ell_k + (i-1)$ may go outside the range $[g_{s_k-1}+1, g_{s_k}]$ for some k. So we use the `mod` operation to ensure that it stays within this range. More formally, we define

$$T_{v_j^i} := \{(\ell_1 + i - 1) \texttt{ mod } I_{s_1}, \ldots, (\ell_r + i - 1) \texttt{ mod } I_{s_r}\},$$

where I_k denotes the interval $[g_{k-1} + 1, g_k + 1]$. This completes the description of the reduction. Note that the reduction takes only polynomial time in the size

of the SetCover instance. We now show that there is a constant factor gap between the optimal values of the Orient-MTW instances created from YES and NO instances of SetCover.

Analysis of YES-instances: We start by showing that in the YES-case, the resulting Orient-MTW instance always admits a solution that visits all leaf nodes.

LEMMA 3.1. *Suppose the* Orient-MTW *instance above is created from a YES-instance* $\mathcal{I} = \{(U, \mathcal{S}), K\}$ *of* SetCover. *Then there is a tour that starts at the root at time 0, and visits all the leaf nodes.*

Proof. By Theorem 2.1, since $\mathcal{I}$ is a YES-instance, there exists a set cover, say, $S_{\ell_1}, \ldots S_{\ell_K}$, where S_{ℓ_j} belongs to $\mathcal{G}_j$ for $j = 1, \ldots, K$. Let Δ denote the index set $\{\ell_1, \ldots, \ell_K\}$. Define Δ_i as the index set Δ *shifted* by $(i-1)$ units, i.e., $\{(\ell_1 + i - 1) \bmod I_1, \ldots, (\ell_K + i - 1) \bmod I_K\}$. First notice that the sets Δ_i are pairwise disjoint for $i = 1, \ldots, g$. Indeed, for any number x, $x \bmod I_k, (x+1) \bmod I_k, \ldots, (x+g-1) \bmod I_k$, are distinct because the length of I_k is $g_k - g_{k-1} + 1 = g + 1$. Consider time $t = (\ell_k + i - 1) \bmod I_k \in \Delta_i$. At time t, the tour will visit all the children v_j^i of w^i which are covered by S_{ℓ_k}, i.e., $\{v_j^i : j \in S_{\ell_k}\}$. Notice that the allowed time slots for these vertices contain time t. Also the edge lengths of the subtree below w^i are 0, and so all these vertices can be visited if the tour happens to be at w^i at time t. Since the distance between w^i and $w^{i'}$ for distinct indices i and i' is 1, we can always ensure that the tour is in the correct sub-tree at each time (also the timeslots in the sets T_v for any leaf vertex are at least 1, and the path from the root to any leaf has length 1/2). Furthermore, the fact that the sets $S_{\ell_1}, \ldots, S_{\ell_K}$ cover all the elements of U implies the tour will visit all the leaf nodes. ∎

Analysis of NO-instances: We now show that in the NO-case, any tour in the resulting Orient-MTW instance fails to visit a constant fraction of the leaf nodes. We start by showing a simple property of any tour. Let G^i denote the sub-tree rooted below w^i.

CLAIM 3.1. *For any integer time* t, *the set of leaf nodes which get visited by any tour at time* t *lie in a particular sub-tree* G^i. *Further, these leaf nodes correspond to the elements contained in one of the sets in* $\mathcal{S}$.

Proof. Fix an integer time t and suppose v_j^i and $v_{j'}^{i'}$ are two vertices which can be visited at time t (i.e., t belongs to $T_{v_j^i}$ and $T_{v_{j'}^{i'}}$). Since both these vertices are getting visited at the same time, distance between them must be 0. It follows that $i = i'$. By definition of the sets $T_{v_j^i}$ and $T_{v_{j'}^{i'}}$, t must be of the form $(\ell_u + i - 1) \bmod I_u$ and $(\ell_r + i - 1) \bmod I_r$, where $S_{\ell_u} \in \mathcal{G}_u$ and $S_{\ell_r} \in \mathcal{G}_r$ are sets containing v_j and $v_{j'}$ respectively. It follows that $T_{v_j^i}$ and $T_{v_{j'}^{i'}}$ lie in the range $[g_{u-1} + 1, g_u]$ and $[g_{r-1} + 1, g_r]$ respectively. These two intervals are disjoint unless $r = u$. But then $\ell_u = \ell_r$, proving the claim. ∎

LEMMA 3.2. *Suppose the* Orient-MTW *instance* $\mathcal{I}'$ *above is created from a NO-instance* $\mathcal{I} = \{(U, \mathcal{S}), K\}$ *of* SetCover. *Then any tour in the instance* $\mathcal{I}'$ *can visit at most* $(1 - 1/e^{80} + o(1))$*-fraction of the leaf nodes.*

Proof. By Theorem 2.1, since $\mathcal{I}$ is a NO-instance, any collection of $c \cdot K$ sets in $\mathcal{S}$ covers at most $(1 - 1/e^{80c})$ elements in U, where c is any positive real number lying in the range $(0, o(\log n)]$. Consider any tour $\mathcal{T}$. At any integer time t, it can only be in one of the subtrees G^i (because all edges incident with the root have positive length). Let A_i denote the integer time instants at which it is in G^i, and let a_i be such that $|A_i| = a_i K$. Since the maximum time by which a vertex can be visited is at most m (recall that m denotes the number of sets in $\mathcal{I}$, and is equal to $g \cdot K$), we get

$$\sum_{i=1}^{g} a_i \leq g. \tag{3.1}$$

Let Z be the collection of sub-trees G^i such that $|a_i| \geq \gamma$ where $\gamma \in [1, o(\log n)]$ is a parameter to be specified later. The above inequality implies that $|Z| \leq g/\gamma$. Consider a sub-tree G^i which is not in Z. Claim 3.1 shows that any time the set of leaves in G^i which get visited during a time $t \in A_i$ correspond to the set of elements in a set in $\mathcal{S}$. Therefore, the assumption in the lemma implies that the tour will cover at most $\left(1 - 1/e^{80 a_i}\right) n$ vertices in G^i, where n denotes $|U|$. Therefore, the number of vertices covered by the tour $\mathcal{T}$ is at most

$$|Z| \cdot n + n \cdot \sum_{i=1}^{g} \left(1 - 1/e^{80 a_i}\right)$$

The second term is maximized when all the a_i's are equal, i.e., are equal to 1 (by inequality (3.1)). Therefore the above expression is at most $gn/\gamma + gn \cdot \left(1 - 1/e^{80}\right)$. Setting $\gamma = \omega(1)$ implies the result as the total number of leaf nodes is gn. ∎

Combining the above two Lemmas with Theorem 2.1, we get

THEOREM 3.1. *There exists a constant* $c > 1$ *such that there is no polynomial time* c*-approximation for the* Orient-MTW *problem, unless* $NP \subseteq DTIME(n^{O(\log \log n)})$.

4 An $\Omega\left(\frac{\log\log n}{\log\log\log n}\right)$-Hardness for Orient-MTW on Trees

In this section, we extend the above construction to amplify the hardness of approximation. We begin by giving some intuition about the construction. Let G be the 2-level tree constructed in the previous section. Also, assume that we are in the "YES" case, and so let $\mathcal{T}$ be a tour which visits all the leaf nodes of G. We first show how to add two more levels to this tree. For each of the leaf nodes v_j^i in G, we will add another copy of G below it – this copy will be a "scaled down" version of G, i.e., the edge length from v_j^i to its children will be scaled down to $1/n^2$ (say) instead of $1/2$. Let $G(v_j^i)$ denote this sub-tree below v_j^i. Let $L(v_j^i)$ be the leaves of $G(v_j^i)$ and L denote the set of leaves of this tree (i.e., $\cup_{i,j} L(v_j^i)$). The tour $\mathcal{T}$ will be extended as follows: suppose the tour visits v_j^i at time t and then visits $v_{j'}^{i'}$ next. In this larger tree, after visiting v_j^i, the tour will first take a detour and visit all the leaves in $L(v_j^i)$, and then come back to v_j^i. After this, it will continue to $v_{j'}^{i'}$, visit the leaves in $L(v_{j'}^{i'})$ and so on. Several issues arise:

- The tour $\mathcal{T}$ may be visiting both v_j^i and $v_{j'}^{i'}$ at the same time t. Now that we have added a detour at v_j^i, we should slightly shift the times at which $v_{j'}^{i'}$ can be visited. This shift would turn out to be $O(1/n)$.
- How should we define T_v for a leaf node v in $L(v_j^i)$? Since the tree $G(v_j^i)$ is a (scaled down) copy of G, the leaf nodes in $G(v_j^i)$ can also be labelled $u_{j''}^{i''}$ where u is an element of U. The construction of G would have allowed a set of time slots $T_{u_{j''}^{i''}}$ for this vertex. Since we are now in a scaled down version of G, the allowed time slots for it should be $O(1/n^2)\cdot T_{u_{j''}^{i''}}$. Since we started from the root of v_j^i of $G(v_j^i)$ at time t, we should shift this by t units. Further, t could be any time in $T_{v_j^i}$ and so we should allow all such possibilities for t while defining the allowed time slots for these leaf vertices.

We now formulate the above concerns below by refining the definition of the graph G. We will parameterize it by two quantities: the lengths of the edges below the root, and the set of timeslots at which we can start at the root (note that the root here could be any vertex v_j^i as described above).

4.1 Refinements of G

Given an instance $\mathcal{I}$ of the SetCover problem, let G be the 2-level tree constructed in the instance $\mathcal{I}'$ of Orient-MTW in the previous section. The leaves of G are labelled v_j^i and every such leaf v_j^i has a set $T_{v_j^i}$ of time slots associated with it. We will now construct a related instance consisting of the same tree G but with two additional parameters – a positive real number Δ and a set Θ of time slots (think of Θ as a collection of times when we can start the tour from the root, earlier this set consisted of time 0 only). Any two distinct times in Θ will differ by more than $\Omega(\Delta\cdot n)$. The intuition for this is the following – the time taken to visit all the leaf nodes in this tree should be much smaller than the difference between any distinct time steps in Θ (when one can start from the root). We shall denote this tree as $G(\Delta,\Theta)$. As before, we shall use r to denote the root of this tree, its children by w^i, and children of w^i by v_j^i. We will continue to use G to denote the "unscaled" version, i.e., where the length of the edges from the root to its children are $1/2$, and $T_{v_j^i}$ be as defined in the previous section.

The tree $G(\Delta,\Theta)$ will have the following properties:

- Each edge from the root r to its children w^i has length $\Delta/4$. Since we are scaling the lengths down by a factor $\Delta/2$, we should define the time slots analogously. In other words, we should define the set of allowable time slots for a leaf vertex v_j^i as $\Delta\cdot T_{v_j^i} := \{\Delta\cdot t : t \in T_{v_j^i}\}$. But as mentioned above, we would like to shift these slightly. So we define $T_{v_j^i}(\Delta) := \Delta\cdot(T_{v_j^i} + \frac{j}{4n}) := \{\Delta\cdot(t+\frac{j}{4n}) : t \in T_{v_j^i}\}$. Note that the maximum allowable time slot for any vertex is at most $m\Delta + \Delta \le 2m\Delta$, where m is the number of sets in the SetCover instance $\mathcal{I}$. We shall use M_Δ to denote $2m\Delta$ – note that any tour of this tree will take at most M_Δ time.
- In the above definition of $T_v(\Delta)$ for a leaf vertex v, we had assumed that we start at the root at time 0. But now, we are given a set of time slots Θ, which denotes the set of possible starting times at r. We will assume that any two times in Θ differ by at least $2M_\Delta$. We define the set of allowable time slots at a leaf vertex v as

$$T_v(\Delta,\Theta) := \cup_{t\in\Theta}(t + T_v(\Delta)),$$

where $t + X$ for a set X denotes $\{t + x : x \in X\}$.

This completes the description of the tree $G(\Delta,\Theta)$. We say that a tour *visits* a node v if it visits it at a time in one of the allowed timeslots for v (i.e., in $T_v(\Delta,\Theta)$). Having defined these refinements, we note some easy to verify properties. The first property says that any time in $T_v(\Delta,\Theta)$ can be uniquely associated with a time $t \in \Theta$. The second property shows that for any tour, the set of vertices which get visited during a window of

length Δ can be associated with the elements of a set in $\mathcal{S}$.

CLAIM 4.1. *The instance $G(\Delta, \Theta)$ has the following properties:*

- *For any leaf v, each time in $T_v(\Delta, \Theta)$ lies in the range $[t + s\Delta, t + (s + 1/4)\Delta]$ for a unique integer s and a unique time $t \in \Theta$.*
- *For an integer s, time $t \in \Theta$ and tour $\mathcal{T}$, the set of leaf vertices visited by it during $[t + s\Delta, t + (s + 1)\Delta)$ have a common parent w^i. Furthermore, the children of w^i visited by this tour correspond to the elements contained in one of the sets in $\mathcal{S}$.*

Proof. Consider a leaf $v := v^i_j$ and $t_v \in T_v(\Delta, \Theta)$. Since any element in $T_v(\Delta)$ is at most M_Δ, and any two times in Θ are separated by at least $2M_\Delta$, we can find a unique $t \in \Theta$ such that $t_v = t + t'$, where $t' \in T_v(\Delta)$. But any time in $T_v(\Delta)$ is of the form $\Delta \cdot (s + \frac{j}{4n})$, where s is a positive integer. Since $j \leq n$, t_v uniquely determines s as well. This proves the first part.

For the second statement, observe that these leaf vertices must be visited during $[t + s\Delta, t + (s + 1/4)\Delta]$ (using the first part above). If v_1 is a child of w^i and v_2 is a child of w^j with $i \neq j$, then the distance between v_1 an v_2 is at least $\Delta/2$. So both cannot be visited during this interval. To complete the proof of the second statement, notice that the children v^i_j of w^i which get visited by this tour during this time window must satisfy the property that $s \in T_{v^i_j}$. The claim now follows as in the proof of Claim 3.1. ∎

The proof of the following is similar to that of Lemma 3.1.

LEMMA 4.1. *Consider the instance $G(\Delta, \Theta)$. If the "YES" case holds, then given any time $t \in \Theta$, there is a tour $\mathcal{T}$ which starts at the root at time t, visits all the leaf nodes in the tree and comes back to the root by time $t + M_\Delta$. The tour $\mathcal{T}$ also has the following property: let v_1 and v_2 be two consecutive leaf vertices on this tour which are visited at time t_1 and t_2 respectively. Let $d(v_1, v_2)$ be the distance between v_1 and v_2 in the tree. Then $(t_2 - t_1) - d(v_2, v_1) \geq \frac{\Delta}{4n}$.*

Proof. The proof of the existence of a tour which visits all the leaf vertices is same as that of Lemma 3.1. Let v^i_j and $v^{i'}_{j'}$ be two consecutive vertices on this tour. If $i = i'$, then the distance between them is 0. But their visit times will be separated by at least $\frac{\Delta}{4n}$. If $i \neq i'$, then Claim 4.1 implies that their visit times are separated by at least $\frac{3\Delta}{4}$. But the distance between them is only $\frac{\Delta}{2}$. This proves the completeness property.

∎

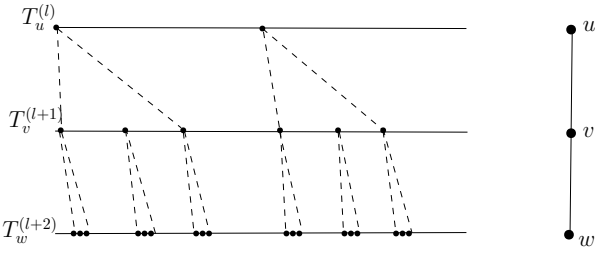

Figure 2: u, v, w are vertices in $V^{(l)}, V^{(l+1)}, V^{(l+2)}$ respectively for an integer l, and v is a (level-2) descendant of u (and similarly for w). The dotted lines show that the times in $T_v^{(l+1)}$ (or $T_w^{(l+2)}$) can be clustered into groups, where each group is obtained by shifts on a unique time in $T_u^{(l)}$ (or $T_v^{(l+1)}$)

4.2 The reduction Armed with Lemma 4.1, we now show the reduction from the set cover instance $\mathcal{I}$ to an instance $\mathcal{I}''$ of the Orient-MTW problem on trees that gives us an $\Omega(\log \log N)$-hardness where N denotes the size of our final instance. The tree H in $\mathcal{I}''$ will be constructed in stages, with each successive stage creating a tree of higher depth that further amplifies the hardness gap. We shall use $H^{(l)}$ to denote the tree obtained after l iterations. Also, to avoid confusion in notation, we shall use $T_v^{(l)}$ to denote the time windows for a leaf vertex v of $H^{(l)}$.

Initially, for $l = 0$, the tree $H^{(0)}$ consists of a single (root) node r with $T_r^{(0)} := \{0\}$. Let ε be a small enough parameter ($\varepsilon = \frac{1}{n^2 g}$ will suffice). Let Δ_l denote ε^l. Suppose we have defined $H^{(l)}$. The tree $H^{(l+1)}$ is constructed as follows: for each leaf node $v \in H^{(l)}$, we attach a copy of the graph $G(\Delta_l, \Theta_v)$ below it with the set Θ_v for the "root" v defined as $T_v^{(l)}$. This also defines the set $T_w^{(l+1)}$ for a leaf vertex of $H^{(l+1)}$, i.e., $T_w(\Delta_l, \Theta_v)$, where v is the leaf vertex in $H^{(l)}$ which is an ancestor of w. We will have η stages, where η will be $\Theta(\log n)$. Let $H := H^{(\eta)}$ denote the final tree. Even though we have defined timeslots for every vertex, we will assign profit 0 to all non-leaf vertices of H, and profit 1 to all leaf vertices.[1]

Recall that M_Δ represents $2m\Delta$. For sake of brevity, we shall use M_l to denote M_{Δ_l}. We have chosen ε small enough so that $M_l \geq \Delta_l \gg M_{l+1}$. Observe that the quantities M_l decrease by a factor ε as l increases (also see Figure 2 for a typical scenario). We shall use $V^{(l)}$ to denote the leaf vertices in $H^{(l)}$. First, a simple observation. Recall that for a vertex $v \in V^{(l)}$, the definition of Θ_v (when we attach a copy of $G(\Delta_l, \Theta_v)$

[1] Even though the problem formulation only counts the number of vertices visited, we can reduce from this profit version by defining the allowable timeslots to $\emptyset$ for all profit 0 vertices.

below it) required that any two times in it differ by at least $2M_l$. We show this property below (observe that Θ_v is same as $T_v^{(l)}$).

CLAIM 4.2. *Let v be a vertex in $V^{(l)}$. Then any two distinct times in $T_v^{(l)}$ are at least $\Delta_{l-1} \geq 2M_l$ apart.*

Proof. The proof is by induction on l. The base case for the root follows vacuously. So assume that the statement is true for l. Let v be a leaf node in $H^{(l)}$, and w be a leaf node in $H^{(l+1)}$ which is contained in the level-2 sub-tree below v, and let $t_1, t_2 \in T_w^{(l+1)}$. We know that $t_1 = t_1' + c_1\Delta_l + \frac{j\Delta_l}{4n}$ and $t_2 = t_2' + c_2\Delta_l + \frac{j\Delta_l}{4n}$, where $t_1', t_2' \in T_v^{(l)}$ and c_1, c_1', j are positive integers. First assume $t_1' = t_2'$. Then it must be the case that $c_1 \neq c_2$. So $|t_1 - t_2| \geq \Delta_l \geq 2M_{l+1}$, assuming ε is small enough.

If $t_1' \neq t_2'$, we know by induction hypothesis that $|t_1' - t_2'| \geq \Delta_{l-1}$. Assume wlog that $t_1' < t_2'$. We know that $t_1 \in [t_1', t_1' + M_l]$. Therefore, $t_2 - t_1 \geq t_2 - t_1' \geq \Delta_{l-1} - M_l \geq \Delta_l$.

∎

Completeness Property. We are now ready to prove the completeness guarantee.

THEOREM 4.1. *Suppose the instance $\mathcal{I}$ satisfies the "YES" case property. Then there is a tour which starts at the root at time 0 and visits all the leaves of H.*

Proof. We prove the following by induction on l: let v be a leaf vertex in $H^{(l)}$ and t be a time in $T_v^{(l)}$. Then there is a tour which starts from v at time t, visits all the leaf vertices in the sub-tree rooted at v, and comes back to v by time $t + \frac{\Delta_{l-1}}{4n}$.

The base case when $l = \eta$ is easy: a vertex $v \in H^{(l)}$ is a leaf vertex, and so the tour just needs to stay at v. Assume it is true for $l+1$. Let v be a leaf vertex in $H^{(l)}$ and $t \in T_v^{(l)}$. Let X be the 2-level sub-tree below v. By Lemma 4.1, there is a tour $\mathcal{T}$ which starts at time t in v, visits all the leaf nodes in X and comes back to v by time $t + M_l$. For each leaf vertex w of X, we perform the following detour of $\mathcal{T}$ – when the tour visits w (at a time $t_w \in T_w^{(l+1)}$), we use induction hypothesis to take a detour from w to visit all the leaf nodes in the sub-tree rooted at w, and then continue back to $\mathcal{T}$ beyond w. To see that this tour is still feasible, notice that each such detour will take an extra $\Delta_l/4n$ amount of time (by induction hypothesis). But by Lemma 4.1, we have a "slack" of $\Delta_l/4n$ between any two consecutive vertices in $\mathcal{T}$.

Finally, note that this tour will be back to v by time $t + M_l \leq t + \frac{\Delta_{l-1}}{4n}$, assuming ε is small enough. Applying this to the root vertex yields the desired statement. ∎

Soundness Property. We now come to the more difficult soundness property. We would like to formalize the intuition in Figure 2. First some definitions. For a node v, let $\mathsf{tree}(v)$ denote the sub-tree of H rooted at v, and $\mathsf{leaf}(v)$ denote the set of leaf nodes in $\mathsf{tree}(v)$. For an index l, let $M_{\geq l}$ denote $M_l + M_{l+1} + \ldots + M_{\eta-1}$. Since the terms in this sequence are in a geometrically decreasing sequence, it is easy to check that $M_{\geq l} \leq 2M_l$. For each vertex $v \in V^{(l)}$ (recall that $V^{(l)}$ denotes the set of leaf vertices in $H^{(l)}$), we shall define two types of time windows associated with it. For a time $t \in T_v^{(l)}$, define $\mathsf{TimeW}_v(t)$ to be the time window $[t, t + 2M_l]$. Claim 4.2 implies that any two such time windows for distinct times in $T_v^{(l)}$ are disjoint. In fact, we show below that any node in $\mathsf{leaf}(v)$ can be visited during such time windows only; we defer the proof to Appendix A.

CLAIM 4.3. *Let u be a leaf vertex in H and let v be the ancestor of u in $V^{(l)}$. For every time $t_u \in T_u^{(\eta)}$, there is a unique time $t_v \in T_v^{(l)}$ such that $\mathsf{TimeW}_v(t_v)$ contains t_u.*

We now define the second kind of time window associated with a vertex $v \in V^{(l)}$. Given a time $t \in T_v^{(l)}$, and integer $c \geq 0$, define $\mathsf{RefinedTimeW}_v(t, c)$ to be the time window $[t + c\Delta_l, t + (c + \frac{1}{2})\Delta_l]$. Again it is not hard to show that these are mutually disjoint for different pairs (t, c) (for a fixed v), and that these are all contained in $\mathsf{TimeW}_v(t)$, provided $c \leq gK$. We have the following refinement of Claim 4.3, proved in Appendix A.

COROLLARY 4.1. *Let $v \in V^{(l)}$ and $u \in \mathsf{leaf}(v)$. Let $w \in V^{(l+1)}$ be the ancestor of u in $V^{(l+1)}$ (so w lies on the v-u path). For every time $t_u \in T_u^{(\eta)}$, there are unique times $t_v \in T_v^{(l)}$, $t_w \in T_w^{(l+1)}$ and positive integer c, $1 \leq c \leq gK$, such that such that $t_u \in \mathsf{TimeW}_w(t_w) \subseteq \mathsf{RefinedTimeW}_v(t_v, c)$*

The following claim further develops the relations between the two kinds of time windows; its proof is also deferred to Appendix A.

CLAIM 4.4. *Let $v \in V^{(l)}$ and $w \in V^{(l+1)}$ be a descendant of v. For an integer c, $1 \leq c \leq gK$ and time $t_v \in T_v^{(l)}$, there is at most one time $t_w \in T_w^{(l+1)}$ such that $\mathsf{TimeW}_w(t_w) \cap \mathsf{RefinedTimeW}_v(t_v, c)$ is non-empty. Further, if this intersection is non-empty, then $\mathsf{TimeW}_w(t_w) \subseteq \mathsf{RefinedTimeW}_v(t_v, c)$*

Having shown these preliminary results, we are ready to show the soundness part of the hardness result. We define a collection of functions $g_l(x)$, $0 \leq l \leq \eta$, for non-negative values of x as follows:

DEFINITION 4.1. *The function $g_\eta(x) = 1$ for all positive x. The function g_l is inductively defined as follows (define $g_l(0)$ as 0):*

$$g_l(x) := \left(1 - e^{-80x}\right) g_{l+1}\left(\frac{x}{1 - e^{-80x}}\right)$$

We shall prove the following fact later in the appendix Section B.

THEOREM 4.2. *For all non-negative integer l, the function $g_l(x)$ is monotonically increasing (with x) and concave. Further, $g_0(1)$ is $O\left(\frac{1}{\log \eta}\right)$.*

Let $v \in V^{(l)}$ and x be an integer. We say that a tour $\mathcal{T}$ *enters* v at most x times if there are x times in $T_v^{(l)}$, say $t_1, \ldots, t_x$ such that the times at which $\mathcal{T}$ visits the leaves in $\mathsf{leaf}(v)$ are contained in $\cup_{i=1}^{x} \mathsf{TimeW}_v(t_i)$ (the existence of such a subset of $T_v^{(l)}$ is guaranteed by Claim 4.3). Define $\mathsf{Max}(v, x)$ as the maximum fraction of leaves in $\mathsf{leaf}(v)$ which can be visited by any tour which enters v at most x times. Finally, define

$$f_l(x) := \max_{v \in V^{(l)}} \mathsf{Max}(v, x).$$

We now prove the main technical result of this section:

THEOREM 4.3. *For all non-negative integers x and integer l, $0 \le l \le \eta$,*

$$f_l(x) \le g_l(x) + \frac{Cx(\eta - l)}{\log n},$$

where C is the constant appearing the statement of Theorem 2.1.

Proof. The proof is by reverse induction on l. Base case for $l = \eta$ is easy – if we are at a leaf node v, $\mathsf{Max}(v, x) = 1$ for all positive integers x. We now show the induction step. Suppose the above statement is true for $l + 1$. Fix a vertex $v \in V^{(l)}$ and positive integer x. Also fix a tour $\mathcal{T}$ which enters v at most x times. Let L be the set of leaves in $\mathsf{leaf}(v)$ which are visited by $\mathcal{T}$. Let $T \subseteq T_v^{(l)}$, $|T| = x$, be the subset of times such that all the vertices in L get visited during $\cup_{t \in T} \mathsf{TimeW}_v(t)$. Corollary 4.1 shows that these leaves must be visited during $\mathsf{RefinedTimeW}_v(t, c)$ for some integer c, $1 \le c \le gK$, and time $t \in T$. Let Ind denote the index set $\{(t, c) | t \in T, 1 \le c \le gK\}$. Clearly, $|Ind| = xgK$.

In order to apply the induction hypothesis, we look at the level-2 subtree below v. Let $w^1, \ldots, w^g$ be the children of v. Recall that each node w^i has n children – call these $v_j^i, j = 1, \ldots, n$. For a node u, let $L(u)$ denote $L \cap \mathsf{leaf}(u)$. Similarly, for a node u, let $\mathsf{Ind}(u)$ denote the set of pairs (t, c) such that all the leaves in $L(u)$ are visited during $\cup_{(t,c) \in \mathsf{Ind}(u)} \mathsf{RefinedTimeW}_v(t, c)$. The proof of the claim below is very similar to the second statement in Claim 4.1, and is deferred to Appendix A.

CLAIM 4.5. *The sets $\mathsf{Ind}(w^i)$, $i = 1, \ldots, g$, are mutually disjoint.*

Since $|\mathsf{Ind}| = xgK$, the above Claim implies that

$$\sum_{i=1}^{g} x_i \le xgK, \tag{4.2}$$

where $x_i = |\mathsf{Ind}(w^i)|$. We classify the children of v into large and small classes depending on the value of x_i.

DEFINITION 4.2. *We say that the node w^i is* large *if $x_i \ge \frac{K \log n}{C}$, where C is the constant in the statement of Theorem 2.1. We say that w^i is* small *if it is not large.*

Inequality (4.2) implies that there are at most $\frac{xgKC}{K \log n} = \frac{gC}{\log n}$ large nodes. We will assume that the tour visits all the leaves under them, which form a $\frac{C}{\log n}$ fraction of the leaves under v (because v has g children). So from now on we focus on small nodes only.

For a pair $(t, c) \in \mathsf{Ind}$, let $S(t, c)$ be the set v_j^i of level-2 descendants of v for which $(t, c) \in \mathsf{Ind}(v_j^i)$. Claim 4.1 shows that for any index (t, c), all the nodes in $S(t, c)$ will have a common parent w^i.

CLAIM 4.6. *For any index $(t, c) \in \mathsf{Ind}$, there is a set S in the $\mathsf{SetCover}$ instance $\mathcal{I}$ such that $S(t, c) \subseteq S$ (where an element v_j^i in $S(t, c)$ corresponds to the element v_j in $\mathcal{I}$).*

Proof. For every vertex v_j^i, it follows from Claim 4.3 and Corollary 4.1 that there is a time $t_j^i \in \mathsf{RefinedTimeW}_v(t, c) \cap T_{v_j^i}^{(l+1)}$. Since the distance between any two nodes in $S(t, c)$ is 0, there is a tour which visits all the nodes in $S(t, c)$ during $\mathsf{RefinedTimeW}_v(t, c)$. The result now follows from Claim 4.1. ∎

Since every set in the $\mathsf{SetCover}$ instance $\mathcal{I}$ has size n/K, where n is the size of the universe in $\mathcal{I}$, we see that $|S(t, c)| \le n/K$. Let $S(w^i)$ denote the set of children u_j^i of w^i for which $L(u_j^i)$ is non-empty. In other words, $S(w^i) = \cup_{(t,c) \in \mathsf{Ind}(w^i)} S(t, c)$. The statement of Theorem 2.1 (the "NO" case) along with Claim 4.6 implies that

$$|S(w^i)| \le \left(1 - e^{-\frac{80x_i}{K}}\right) n. \tag{4.3}$$

For a node v_j^i, let $x_{i,j}$ denote $|\mathsf{Ind}(v_j^i)|$. Since $|S(t,c)| \leq n/K$, for any pair (t,c), it follows that for each i,

$$\sum_{j:v_j^i \in S(w^i)} x_{i,j} \leq x_i \cdot \frac{n}{K}. \tag{4.4}$$

Claim 4.4 implies that corresponding to each $(t,c) \in \mathsf{Ind}(v_j^i)$, there is a unique time $t' \in T_{v_j^i}^{(l+1)}$ such that $\mathsf{TimeW}_{v_j^i}(t') \subseteq \mathsf{RefinedTimeW}_v(t,c)$ and all the leaves in $L(v_j^i)$ which are visited during $\mathsf{RefinedTimeW}_v(t,c)$ are actually visited during $\mathsf{TimeW}_{v_j^i}(t')$. Therefore, by induction hypothesis,

$$\frac{|L(v_j^i)|}{\mathsf{leaf}(v_j^i)} \leq f_{l+1}(x_{i,j}).$$

It follows that the fraction of leaves in $L(w^i)$ which are visited by the tour is at most (note that $\mathsf{leaf}(v_j^i)$ has the same cardinality for all j)

$$\frac{1}{n} \cdot \sum_{j:v_j^i \in S(w^i)} f_{l+1}(x_{i,j}) \leq \frac{1}{n} \cdot \sum_{j:v_j^i \in S(w^i)} \left(g_{l+1}(x_{i,j}) + \frac{C x_{i,j}(\eta - (l+1))}{\log n} \right).$$

Concavity of g_{l+1}, along with (4.4) imply that the above is at most

$$\frac{|S^i|}{n} g_{l+1}\left(\frac{x_i \cdot n}{K|S_i|}\right) + \frac{C x_i(\eta - (l+1))}{K \log n}.$$

Inequality (4.3) implies that the above is at most

$$\left(1 - e^{-\frac{80x_i}{K}}\right) g_{l+1}\left(\frac{x_i}{K\left(1 - e^{-\frac{80x_i}{K}}\right)}\right) + \frac{C x_i(\eta - (l+1))}{K \log n} = g_l(x_i/K) + \frac{C x_i(\eta - (l+1))}{K \log n}.$$

Inequality (4.2) along with the fact that there are at most $\frac{gC}{\log n}$ large nodes implies that the total fraction of leaves below v which get visited is at most

$$\frac{C}{\log n} + \frac{1}{g} \cdot \left(\sum_{i=1}^{g} g_l(x_i/K) + \frac{C x_i(\eta - (l+1))}{K \log n} \right) \leq \frac{1}{g} \cdot \sum_{i=1}^{g} g_l(x_i/K) + \frac{Cx(\eta - l)}{\log n}.$$

Finally, the concavity of g_l, along with (4.2), implies that the above is at most $g_l(x) + \frac{Cx(\eta-l)}{\log n}$. This proves the induction hypothesis. ∎

Now applying Theorem 4.3 to the root, i.e., $l = 0, x = 1$, shows that any tour will visit only $g_0(1) + \frac{C\eta}{\log n} = O(1/\log \eta)$ fraction of the leaf nodes (Theorem 4.2 and the fact that η is set to $\frac{\log n}{\log \log n}$).

Putting it together: We now analyze the size of the final tree H. It is easy to see that the size of $V^{(l)}$ is $O((ng)^l)$. Since $g \leq n$, the size of the instance H is $N = n^{O(\eta)}$. If we take $\eta = \log n$, we see that $N = n^{O(\log n)}$, and the hardness factor is $\Omega\left(\frac{\log \log N}{\log \log \log N}\right)$. This completes the soundness part of the reduction and proves Theorem 1.1.

5 Conclusion

We have given the first super-constant lower bound on the approximation ratio of a non-trivial version of the vehicle routing problem. However, many interesting problems remain open:

- Can we give a logarithmic polynomial-time approximation algorithm for Orient-MTW on trees ?
- Can we give a constant factor approximation problem for Orient-MTW on trees when all the time windows T_v are of the form $[0, B_v]$ for some input parameters B_v ?
- Can we give a super-constant lower bound (in general graphs) on the approximation ratio of the special case of Orient-MTW when each time window consists of a single interval ? Can we show such a result for directed graphs ?
- In the case of directed graphs, can we give a constant factor approximation algorithms for Orient-MTW when all time windows are of them form $[0, B]$ for a uniform B (the so-called "orienteering" problem) ?

Acknowledgements

The second author was supported in part by NSF awards CCF-1617851, CCF-1763514, CCF-1934876, and CCF-2008305.

References

[1] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, USA, 2007.

[2] Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings*

of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004, pages 166–174, 2004.

[3] Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM J. Comput.*, 37(2):653–670, 2007.

[4] Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Trans. Algorithms*, 8(3):23:1–23:27, 2012.

[5] Chandra Chekuri and Amit Kumar. Maximum coverage problem with group budget constraints and applications. In *Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2004, Cambridge, MA, USA, August 22-24, 2004*, pages 72–83, 2004.

[6] Chandra Chekuri and Martin Pál. A recursive greedy algorithm for walks in directed graphs. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 245–253, 2005.

[7] Ke Chen and Sariel Har-Peled. The euclidean orienteering problem revisited. *SIAM J. Comput.*, 38(1):385–397, 2008.

[8] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633, 2014.

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

[10] Zachary Friggstad and Chaitanya Swamy. Compact, provably-good lps for orienteering and regret-bounded vehicle routing. In *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, pages 199–211, 2017.

[11] Subhash Khot. Hardness of set cover. Lecture Notes, 2020. https://cs.nyu.edu/ khot/PCP-Spring20.html.

[12] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[13] Viswanath Nagarajan and R. Ravi. The directed orienteering problem. *Algorithmica*, 60(4):1017–1030, 2011.

[14] Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998.

[15] Ola Svensson, Jakub Tarnawski, and László A. Végh. A constant-factor approximation algorithm for the asymmetric traveling salesman problem. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 204–213, 2018.

[16] Paolo Toth and Daniele Vigo, editors. *Vehicle Routing*, volume 18 of *MOS-SIAM Series on Optimization*. SIAM, 2014.

A Omitted Proofs from Section 4

Proof of Claim 4.3 The proof is by induction on l. For $l = \eta$, this follows easily (we need to use Claim 4.2 which states that any two times in $T_u^{(\eta)}$ are at least $2M_\eta$ apart). Suppose it is true for $l+1$. Let w be the ancestor of u in $V^{(l+1)}$. By induction hypothesis, there is a unique time $t_w \in T_w^{(l+1)}$ such that $t_u \in [t_w, t_w + 2M_{l+1}]$. Further, the definition of the set $T_w^{(l+1)}$ shows that there is a time $t_v \in T_{v_l}^{(l)}$ such that $t_w \in [t_v, t_v + M_l]$. Combining the two observations, we see that $[t_w, t_w + 2M_{l+1}] \subseteq [t_v, t_v + M_l + 2M_{l+1}] \subseteq [t_v, t_v + 2M_l]$.

Uniqueness of time t_v in the statement follows from the fact that times in $T_v^{(l)}$ are at least $2M_l$ apart (Claim 4.2). □

Proof of Corollary 4.1 By Claim 4.3, there is a unique time $t_w \in T_w^{(l+1)}$ satisfying $t_u \in \mathsf{TimeW}_w(t_w)$. Also, corresponding to t_w, there is a unique $t_v \in T_v^{(l)}$ and a unique positive integer c, $1 \le c \le gK$, such that $t_w \in [t_v + c\Delta_l, t_v + (c + 1/4)\Delta_l]$ (Claim 4.1). The result now follows because $\Delta_l \gg M_{l+1}$. □

Proof of Claim 4.4 Any two times in $T_v^{(l+1)}$ are at least Δ_l apart (by Claim 4.2). So there can be at most one interval time t_w such that $\mathsf{TimeW}_w(t_w) := [t_w, t_w + 2M_{l+1}]$ has non-empty intersection with $\mathsf{RefinedTimeW}_v(t_v, c) := [t_v + c\Delta_l, t_v + (c + \frac{1}{2})\Delta_l]$. By Corollary 4.1, there is a time $t'_v \in T_v^{(l)}$ and integer c' such that $\mathsf{TimeW}_w(t_w)$ is contained in $\mathsf{RefinedTimeW}_v(t'_v, c') := [t'_v + c'\Delta_l, t'_v + (c' + \frac{1}{2})\Delta_l]$. But Claim 3.1 implies that any two distinct times in $T_v^{(l)}$ are at least $2M_l \ge (c + c')\Delta_l$ apart, and so it must be the case that $t_v = t'_v$. But then the fact that $[t_w, t_w + 2M_{l+1}]$ is completely contained in $[t'_v + c'\Delta_l, t'_v + (c' + \frac{1}{2})\Delta_l]$ implies that $c = c'$. □

Proof of Claim 4.5 The proof is very similar to the second statement in Claim 4.1. Indeed, the length of any edge joining v to one of it's children is $\Delta_l/4$, and so, any tour will incur at least $\Delta/2$ time in going from one child of v to another child. Since the length of the interval $\mathsf{RefinedTimeW}_v(t, c)$ is $\Delta_l/2$, the result follows. □

B Proof of Theorem 4.2

For sake of brevity let $h(x)$ denote the function $(1 - e^{-ax})$, where $a = 80$. The following fact about h is easy to check.

Claim B.1. *The function $h(x)$ is increasing function of x and is concave. Further, $h(x) - xh'(x) \ge 0$ for all $x > 0$.*

Here is a more non-trivial property of h:

CLAIM B.2. *The function* $\frac{h(x)-xh'(x)}{h(x)^2}$ *is an increasing function of* x *(for non-negative* x*).*

Proof. Notice that $h''(x) = -ah'(x)$. Using this and differentiating the above expression, we get (suppressing the argument for h)

$$\frac{axh'h - 2h'(h - xh')}{h^3} = \frac{(ax-2)h'h + 2xh'^2}{h^3}.$$

We need to show that the numerator is non-negative for non-negative values of x. Notice that for $x = 0$, the numerator is 0 (because $h(0) = 0$). We will show that the numerator is an increasing function of x, which will imply that it is non-negative for non-negative values of x. Using $h(x) = (1 - e^{-ax})$, the numerator is equal to

$$h'(x)\left((ax-2) + (ax+2)e^{-ax}\right).$$

Since $h'(x) \geq 0$, it is enough to show that $(ax-2)e^{ax} + (ax+2) \geq 0$. But this follows easily from the fact that $e^{ax} \geq (1+ax)$ for non-negative x. ∎

CLAIM B.3. *The function* $g_l(x)$ *is monotonically increasing.*

Proof. Let $p(x)$ denote $\frac{x}{h(x)}$. We differentiate g_l to get (we suppress the arguments for sake of brevity, but note that g_{l+1} is really denoting $g_{l+1}(p(x))$):

$$g_l' = h'g_{l+1} + hg_{l+1}'p' = h'g_{l+1} + \frac{g_{l+1}'(h - xh')}{h}.$$

The first term is non-negative because $h'(x) > 0$ and g_{l+1} is always non-negative. To show that the second term is non-negative, notice that $h(x)$ is always positive and g_{l+1}' is non-negative by induction hypothesis. The result follows by Claim B.1. ∎

We now proceed to show that the function is concave. This will be a more tedious calculation. Fix a positive real x. We define a sequence $x_\eta = x, x_{\eta-1}, \ldots, x_1, x_0$ recursively as follows:

$$x_i = \frac{x_{i+1}}{h(x_{i+1})} \quad i = \eta - 1, \ldots, 0.$$

Note that x_l really is a function of x. So we should be using the notation $x_l(x)$, but we will skip the argument whenever it is clear from the context. The following claim is easy to check.

CLAIM B.4. $x_l(x)$ *is an increasing function of* x*. Further* $x_l'(x)$ *is an increasing function of* x*.*

Proof. Proof is by reverse induction on l. For $l = \eta$, this follows because $x_l(x) = x$. Now suppose it is true for $l+1$. Differentiating the expression for x_l and suppressing the argument x, we get

$$\begin{aligned} x_l' &= \frac{x_{l+1}'}{h(x_{l+1})} - \frac{x_{l+1}h'(x_{l+1})x_{l+1}'}{h(x_{l+1})^2} \\ &= \frac{x_{l+1}'(h(x_{l+1}) - x_{l+1}h'(x_{l+1}))}{h(x_{l+1})^2} \end{aligned}$$

By induction hypothesis, $x_{l+1}' \geq 0$. The first statement in the claim now follows from Claim B.1.

To prove the second statement, we again use induction. Since $x_\eta' = 1$, the base case follows. Now suppose x_{l+1}' is an increasing function of x. The result now follows from the above equation and Claim B.2. ∎

CLAIM B.5. *For all* $l, 0 \leq l \leq \eta$, $g_l(x) = \frac{x}{x_l}$.

Proof. The proof is by induction on l. For $l = \eta$, $g_l(x) = 1$, and so the statement holds. Assume it is true for $l+1$.

$$g_l(x) = h(x)g_{l+1}\left(\frac{x}{h(x)}\right) = h(x)g_{l+1}(x_1) = h(x)\cdot\frac{x_1}{x_{l+1}(x_1)}$$

where we have applied the induction hypothesis on $g_{l+1}(x_1)$. But note that $x_{l+1}(x_1) = x_l(x)$. Since $h(x)x_1 = x$, the result follows. ∎

The above claim implies that $g_l \cdot x_l = x$. Differentiating this twice, we get

$$g_l''x_l + x_l''g_l + x_l'g_l' = 0.$$

The last two terms are non-negative by Claim B.3 and Claim B.4. Therefore $g_l'' \leq 0$. This implies that g_l is concave function for non-negative x. To complete the proof of Theorem 4.2, we need to upper bound $g_0(1)$. Let x_l denote $x_l(1)$ for rest of the discussion. Claim B.5 implies that $g_l(1)x_l = 1$. So $g_0(1)x_0 = 1$. Let η' be such that $\eta = \eta' \log \eta'$. We will show that $x_0 \geq \log \eta'$, which will imply the desired result. Suppose, for the sake of contradiction, this is false. Since $x_i \geq x_{i+1}$, this implies that $x_l \leq \eta'$ for all $0 \leq l \leq \eta$. But then

$$x_l = \frac{x_{l+1}}{h(x_{l+1})} \geq \frac{x_{l+1}}{1 - 1/\eta'}.$$

Therefore,

$$x_0 \geq \frac{1}{(1-1/\eta')^\eta} \geq \eta',$$

which is a contradiction.

C Proof of Theorem 2.1

In this section, we give details of the reduction from 3-SAT to SetCover which has the desired properties mentioned in the theorem statement. We begin by stating the reduction from 3-SAT to Label Cover problem, and then give the reduction from Label Cover to SetCover.

An instance $\mathcal{L}$ of Label Cover is given by a tuple $(G = (V, W, E), [M], [N], \pi)$, where G is a bipartite graph with V, W on the two sides, E is the set of edges and $\pi : [M] \to [N]$ is a projection map. A vertex in V can be labelled from an element in $[N]$, whereas as vertex in W gets label from $[M]$. A labeling $\ell : V \to [N], W \to [M]$ is said to satisfy an edge $e = (v, w)$ if $\ell(v) = \pi(\ell(w))$.

The following is a consequence of the well-known Parallel Repetition theorem [14].

THEOREM C.1. *Given a 3-SAT instance ϕ of size n and a parameter k, we can construct an instance $\mathcal{L} = (G = (V, W, E), [M], [N], \pi)$ of label-cover with the following properties:*

- *If ϕ is satisfiable, then there is a labeling in $\mathcal{L}$ which satisfies all the edges.*
- *If ϕ is not satisfiable, then any labeling in $\mathcal{L}$ satisfies at most $2^{-\alpha k}$ fraction of edges, where α is a constant.*

Further, $|V| = |W| = n^{O(k)}, |E| = 2^{O(k)}$, $M = 7^k, N = 2^k$ and each vertex has the same degree. This reduction can be carried out in $n^{O(k)}$ time.

Now we give a reduction from Label Cover (as given by the Theorem above) to SetCover. The reduction is the same as given by [12], but we rely on the exposition by Khot [11]. First we construct a suitable set system.

C.1 A partition system A partition system $\mathcal{P} = (U, m, h, t)$ consists of the following:

- A universe U of m elements.
- Pairs of sets $(A_1, \bar{A}_1), \ldots, (A_t, \bar{A}_t)$, with each $A_i \subseteq U$ of size exactly $m/2$.
- For every $h' < h$ and every family of subsets $B_{i_1}, B_{i_2}, \ldots, B_{i_{h'}}$, where each B_{i_l} is either A_{i_l} or $\bar{A}_{i_l}$,
$$|\cup_{\ell=1}^{h'} B_{i_\ell}| \leq m - \frac{m}{2^{h'+2}}.$$

We now show how to construct such a set system efficiently with $m = O(2^h h \log t)$. We pick subsets $A_1, \ldots, A_t$, where each A_i is a random subset of U (by assigning each element to A_i with probability 1/2). Now to prove the property above, fix an $h' \leq h$. Consider a family of h' subsets $B_{i_1}, B_{i_2}, \ldots, B_{i_{h'}}$ as above. The expected number of elements not covered by them is $\frac{m}{2^{h'}} \geq 100h \log t$. Therefore, the probability that they leave out more than $50h \log t$ elements is at least $e^{-10h \log t}$. Taking a union bound over all choices of such h' subsets, we see that the probability of not satisfying the above condition is at most

$$\sum_{h'=1}^{h} \binom{t}{h'} 2^{h'} e^{-10h \log t}$$

which can be made very small for large h, t.

We would like each subset A_i to have size exactly $m/2$. By Chernoff bounds, each A_i is close to $m/2$. So we can keep aside a subset U' of $m/2$ elements and construct the above set system on the remaining $m/2$ elements only. We can then make sure that both the sets $A_i, \bar{A}_i$ have size exactly $m/2$ by suitably assigning the elements of U'. Assume we have a set system $\mathcal{P}$ as above.

C.2 Reduction Given a label cover instance $\mathcal{L}$ with $|V| = |W|$, $M = 7^k, N = 2^k$, we first make $|E|$ independent copies of the set system $\mathcal{P}$ – call these $\mathcal{P}_e = (U_e, m, h, t)$ for each edge $e \in E$. Let the corresponding subsets $A_1, \ldots, A_t$ in $\mathcal{P}$ be labeled $A_1^e, \ldots, A_t^e$ in $\mathcal{P}_e$. Further, we choose $t = 2^k$, so that each subset A_i can be identified with a label $i \in [N]$.

In the SetCover instance $(X, \mathcal{S})$, the ground set X would be $\cup_e U_e$. We define the sets in $\mathcal{S}$ as follows: for every $v \in V, i \in [N]$, we have a set

$$S_{v,i} = \cup_{w:e=(v,w)\in E} A_i^e.$$

Similarly for every $w \in W, j \in [M]$, we have a set

$$S_{w,j} = \cup_{v:e=(v,w)\in E} \bar{A}_{\pi(j)}^e.$$

For any $v \in V$, let $\mathcal{S}(v)$ denote the family $\{S_{v,i} : i \in [N]\}$, and for any $w \in W$, we define $\mathcal{S}(w)$ similarly. Since the degree of each vertex in G is same, and all the sets in $\mathcal{P}$ have the same size, it follows that all the sets in $\mathcal{S}$ have the same size as well. Furthermore, for any $v, v' \in V$, $|\mathcal{S}(v)| = |\mathcal{S}(v')|$, and similarly, for any $w, w' \in W$, $|\mathcal{S}(w)| = |\mathcal{S}(w')$. By arbitrarily repeating sets, as needed, we can further ensure that for any $v \in V$ and $w \in W$, we have $|\mathcal{S}(v)| = |\mathcal{S}(w)|$.

The following lemma follows easily:

LEMMA C.1. *If $\mathcal{L}$ has a labeling which satisfies all the edges, then there is a set cover of the instance $(X, \mathcal{S})$ of size at most $|V| + |W|$. Furthermore, for each $v \in V$, this set cover contains exactly one set from the family $\mathcal{S}(v)$, and similarly for each $w \in W$, this set cover contains exactly one set from the family $\mathcal{S}(w)$.*

Now we consider the more non-trivial direction. Suppose no labeling satisfies more than $2^{-\alpha k}$ fraction of edges in $\mathcal{L}$. Consider a collection of at most $h'(|V|+|W|)$ sets in $\mathcal{S}$ – call this collection $\mathcal{S}'$, where $1 \leq h' \leq h/16$. For a vertex v (or w), let $\mathcal{S}'(v)$ denote $\mathcal{S}(v) \cap \mathcal{S}'$. Let $V_1 \subset V$ be the set of vertices v for which $\mathcal{S}'(v) \leq 8h'$, and define W_1 similarly. We first observe that $|V_1| \geq 3|V|/4$. Indeed, otherwise number of sets in $\mathcal{S}'$ will be more than $\frac{|V|}{4} \cdot 8h' = h'(|V| + |W|)$, which is a contradiction (recall that $|V| = |W|$). Similarly, $|W_1| \geq 3|W|/4$. Let G' be the sub-graph induced by (V_1, W_1), and E' be the set of edges between V_1 and W_1. Since any edge in $E \setminus E'$ has to be incident on either $V \setminus V_1$ or $W \setminus W_1$, it follows that $|E'| \geq |E|/2$.

We call an edge $e = (v, w) \in E'$ *good* if there is a label j such that $\mathcal{S}'(w)$ contains $S_{w,j}$ and $\mathcal{S}'(v)$ contains $S_{v,i}$, where $i = \pi(j)$. We first argue that at most half the edges in E' are good. Suppose not. We will give a labeling of vertices which will satisfy a non-trivial fraction of edges. For each $v \in V$, let $L(v)$ denote the set of labels i such that $S_{v,i} \in \mathcal{S}$ and similarly define $L(w), w \in W$. We pick a random label from each $L(v), v \in V$ and $L(w), w \in W$. A good edge e will be satisfied with probability at least $\frac{1}{64h'^2}$, and so there is a labeling which satisfies at least $\frac{|E'|}{128h'^2} \geq \frac{|E|}{256h'^2}$ edges. We choose h such that $\frac{1}{h^2} \geq 2^{-\alpha k}$, which will give us the desired contradiction, because $h' \leq h/16$. For sake of concreteness, choose h to be $2^{\alpha k/2}$.

Now consider an edge $e \in E'$ which is not good. Since the projection of $\mathcal{S}'(v) \cup \mathcal{S}'(w)$ on U_e contains at most $16h' < h$ sets, and no two of them are complement of each other, we see that we will leave out at least $\frac{1}{2^{16h'+2}}$ fraction of elements in U_e. Since at least $1/4$ fraction of edges in E lie in E' but are not good, we leave out at least

$$\frac{1}{4} \cdot \frac{1}{2^{16h'+2}} \geq \frac{1}{2^{20h'}}$$

fraction of elements in the ground set X (recall that $h' \geq 1$).

Pick k such that $n^k = 2^{2^{\alpha k/2}} = 2^h$. So $k = O(\log \log n)$. The size of the set cover instance is $N = n^{O(k)} \cdot O(2^h h \log t) = n^{O(k)} \cdot O(2^h hk) = n^{O(k)}$. Therefore $h/16 = \frac{\log N}{C}$ for an absolute constant C.

This completes the description from 3-SAT to SetCover. We now show that it has the desired properties. As mentioned above the size of the SetCover instance is $n^{O(\log \log n)}$. The construction of this instance also takes $n^{O(\log \log n)}$ time. As shown above every set in $\mathcal{S}$ has the same size. The parameter K is $|V|+|W|$, and the groups $\mathcal{G}_1, \ldots, \mathcal{G}_K$ correspond to the sets $\mathcal{S}(u), u \in V \cup W$. As shown above, each of these groups $\mathcal{G}_i$ have the same number of sets. In the completeness case, Lemma C.1 shows that there is a set cover containing exactly one set from $\mathcal{G}_i$ for each i.

We now come to the soundness case. As shown above, for every $c \in [1, \log N/C]$, any collection of cK sets in the SetCover instance will leave out at least $\frac{1}{2^{20c}} \leq e^{-20c}$ fraction of elements. It remains to consider the range $c \in (0, 1)$. We consider the following cases:

- $0 < c \leq 1/4$: Since all sets in $\mathcal{S}$ have the same size, any collection of cK sets will leave out at least $(1-c)$-fraction of the elements. Since $(1-c) \geq e^{-4c}$ if $c < 1/4$, the result follows in this case.
- $1/4 \leq c < 1$: We have shown that any collection K sets will leave out at least $1/e^{-20c}$ fraction of the elements. Since $80c \geq 20$, the result follows.

This completes the proof of Theorem 2.1.