A Path-Decomposition Theorem with Applications to Pricing and Covering on Trees[⋆]

Marek Cygan[1], Fabrizio Grandoni[1], Stefano Leonardi[2], Marcin Pilipczuk[3], Piotr Sankowski[3]

[1] IDSIA, University of Lugano, Switzerland, {marek,fabrizio}@idsia.ch
[2] Department of Computer and System Science, Sapienza University of Rome, Italy, leon@dis.uniroma1.it
[3] Institute of Informatics, University of Warsaw, Poland, {malcin,sank}@mimuw.edu.pl

Abstract. In this paper we focus on problems characterized by an input n-node tree and a collection of subpaths. Motivated by the fact that some of these problems admit a very good approximation (or even a poly-time exact algorithm) when the input tree is a path, we develop a decomposition theorem of trees into paths. Our decomposition allows us to partition the input problem into a collection of $O(\log\log n)$ subproblems, where in each subproblem either the input tree is a path or there exists a *hitting* set F of edges such that each path has a non-empty, small intersection with F. When both kinds of subproblems admit constant approximations, our method implies an $O(\log\log n)$ approximation for the original problem.

We illustrate the above technique by considering two natural problems of the mentioned kind, namely Uniform Tree Tollbooth and Unique Tree Coverage. In Uniform Tree Tollbooth each subpath has a budget, where budgets are within a constant factor from each other, and we have to choose non-negative edge prices so that we maximize the total price of subpaths whose budget is not exceeded. In Unique Tree Coverage each subpath has a weight, and the goal is to select a subset X of edges so that we maximize the total weight of subpaths containing exactly one edge of X. We obtain $O(\log\log n)$ approximation algorithms for both problems. The previous best approximations are $O(\log n/\log\log n)$ by Gamzu and Segev [ICALP'10] and $O(\log n)$ by Demaine et al. [SICOMP'08] for the first and second problem, respectively, however both previous results were obtained for much more general problems with arbitrary budgets (weights).

1 Introduction

Several natural graph problems are characterized by an input n-node tree $T = (V, E)$, and a (multi-)set $\mathcal{P} = \{P_1, \ldots, P_m\}$ of subpaths of T (the *requests*).

[⋆] The first two authors are partially supported by the ERC Starting Grant NEWNET 279352, whereas the third and fifth author are partially supported by the ERC Starting Grant PAAl 259515.

Typically one has to make decisions on the edges or nodes of T, and based on these decisions one obtains some profit from each request. Some of the mentioned problems admit a very good approximation (or are even poly-time solvable) when T itself is a path, but get substantially harder in the tree case. In this paper we present a tool to reduce the complexity gap between the path and tree case to $O(\log\log n)$ for problems which have some extra natural properties. We illustrate the application of this method by providing an $O(\log\log n)$ approximation for one pricing problem and for one covering problem, improving on previous best logarithmic approximations.

1.1 Our Results

The main technical contribution of this paper is a simple decomposition theorem of trees into paths.

Theorem 1. *Given an n-node rooted tree T, and a collection of paths $\mathcal{P} = \{P_1, \ldots, P_m\}$, there is a polynomial time algorithm which partitions $\mathcal{P}$ into $\ell = O(\log\log n)$ subsets $\mathcal{P}_1, \ldots, \mathcal{P}_\ell$ such that one of the following two properties holds for each $\mathcal{P}_i$:*

(a) The algorithm provides a partition $\mathcal{L}$ of T into edge disjoint paths, such that each request of $\mathcal{P}_i$ is fully contained in some path of $\mathcal{L}$.
(b) The algorithm provides a collection of edges F_i, such that each $P \in \mathcal{P}_i$ contains at least one and at most 6 edges from F_i.

Intuitively, each subset $\mathcal{P}_i$ naturally induces a subproblem, and at least one of the subproblems carries a fraction $1/O(\log\log n)$ of the original profit. For subproblems of type (a), each L_i induces an independent instance of the problem on a path graph: here the problem becomes typically easier to solve. For subproblems of type (b), a good approximation can be obtained if the objective function and constraints allow us to retrieve a big fraction of the optimal profit from each path P by making a decision among a few options on a small, arbitrary subset of edges of P. In particular, it is sufficient to make decisions (even randomly and obliviously!) on F_i.

We illustrate the above approach by applying it to two natural problems on trees and subpaths. The first one is a pricing problem. In the well-studied Tree Tollbooth problem (TT), we are given a tree $T = (V, E)$ with n vertices, and a set of paths $\mathcal{P} = \{P_1, \ldots, P_m\}$, where P_i has budget $b_i > 0$. The goal is to find a pricing function $p : E \to \mathbb{R}_+$ maximizing $\sum_{1 \le i \le m, p(P_i) \le b_i} p(P_i)$, where $p(P_i) := \sum_{e \in P_i} p(e)$. Here we consider the Uniform Tree Tollbooth problem (UTT), which is the special case where the ratio of the largest to smallest budget is bounded by a constant. Interestingly enough, the best-known approximation for UTT is the same as for TT, i.e. $O(\log n / \log\log n)$ [7]. In this paper we obtain the following improved result.

Theorem 2. *There is an $O(\log\log n)$-approximation for the* Uniform Tree Tollbooth *problem.*

The second application is to a covering problem. In the Unique Coverage on Trees problem (UCT) we are given a tree T and subpaths $\mathcal{P} = \{P_1, \ldots, P_m\}$ as in TT, plus a profit function $p : \mathcal{P} \to \mathbb{R}_+$. The goal is to compute a subset of edges $X \subseteq E$ so that we maximize the total profit of paths $P \in \mathcal{P}$ which share exactly one edge with X.

Theorem 3. *There is an $O(\log \log n)$-approximation for the* Unique Coverage on Trees *problem.*

1.2 Related Work

The tollbooth problem belongs to a wider family of *pricing* problems, which attracted a lot of attention in the last few years. In the *single-minded* version of these problems, we are given a collection of m clients and n item types. Each client wishes to buy a bundle of items provided that the total price of the bundle does not exceed her budget. Our goal is to choose item prices so that the total profit is maximized. In the *unlimited supply* model, there is an unbounded amount of copies of each item. For this problem an $O(\log n + \log m)$ approximation is given in [9]. This bound was refined in [3] to $O(\log L + \log B)$, where L denotes the maximum number of items in a bundle and B the maximum number of bundles containing a given item. A $O(L)$ approximation is given in [1]. On the negative side, Demaine et al. [4] show that this problem is hard to approximate within $\log^d n$, for some $d > 0$, assuming that $NP \not\subseteq BPTIME(2^{n^\varepsilon})$ for some $\varepsilon > 0$.

An interesting special case is when each bundle contains $k = O(1)$ items. The case $k = 2$ is also know as the Vertex Pricing problem. Vertex Pricing is APX-hard even on bipartite graphs [5], and it is hard to approximate better than a factor 2 under the Unique Game Conjecture [12]. On the positive side, there exists a 4-approximation for Vertex Pricing, which generalizes to an $O(k)$-approximation for bundles of size k [1].

A $O(\log n)$ approximation for the Tree Tollbooth problem was developed in [5]. This was recently improved to $O(\log n / \log \log n)$ by Gamzu and Segev [7]. Tree Tollbooth is APX-hard [9]. One might consider a generalization of the problems on arbitrary graphs (rather than just trees). In that case the problem is APX-hard even when the graph has bounded degree, the paths have constant length and each edge belongs to a constant number of paths [3].

The Highway problem is the special case of Tree Tollbooth where the input graph is a path. It was shown to be weakly NP-hard by Briest and Krysta [3], and strongly NP-hard by Elbassioni, Raman, Ray, and Sitters [5]. Balcan and Blum [1] give an $O(\log n)$ approximation for the problem. The result in [7] implies as a special case an $O(\log n / \log \log n)$ for the problem. Elbassioni, Sitters, and Zhang [6] developed a QPTAS, exploiting the profiling technique introduced by Bansal et al. [2]. Finally, a PTAS was given by Grandoni and Rothvoß [8]. FPTASs are known for some special case: for example when the graph has constant length [11], the budgets are upper bounded by a constant [9], the paths have constant length [9] or they induce a laminar family [1, 3]. We will (crucially) use the PTAS in [8] as a black box to derive an $O(\log\log n)$ approximation for Uniform Tree Tollbooth.

One can also consider variants of the above problems. For example, in the *coupon* version, prices can be negative and the profit of a bundle is zero if its total price is above the budget or below zero. The coupon version is typically harder than the classical one. For example, Coupon Highway is APX-hard [5]. Recently Wu and Popat [13] showed that Coupon Highway and Coupon Vertex Pricing are not approximable within any constant factor given the Unique Game Conjecture.

Versions of the above problems with a limited supply are much less studied. Here, part of the problem is to assign the available copies to the clients who can afford to pay them. In the latter case, one can consider the *envy-free* version of the problem, where prices must satisfy the condition that all the clients who can afford the price of their bundle actually get it [9].

Unique Coverage on Trees is a special case of the Unique Coverage problem: given a universe U a collection $\mathcal{F} \subseteq 2^U$ of subsets of U, the goal is to find a subset of the universe $X \subseteq U$, such that the number of sets in $\mathcal{F}$ containing exactly one element of X is maximized. Demaine et al. [4] show that this problem is hard to approximate within $\log^d n$, for some $d > 0$, assuming that $NP \not\subseteq BPTIME(2^{n^\varepsilon})$ for some $\varepsilon > 0$. However, if there is a solution covering all the sets in $\mathcal{F}$, then an e-approximation is known [10].

2 A Decomposition Theorem

In this section we prove Theorem 1. In order to introduce ideas in a smooth way, in Section 2.1 we start by proving a simplified version of the theorem in the case that the input graph has small (namely, logarithmic) diameter. In Section 2.2 we generalize the approach to arbitrary diameters.

For a positive integer a, by $\texttt{POWER}_2(a)$ we denote the smallest integer i such that 2^i does not divide a; in other words, $\texttt{POWER}_2(a)$ is the index of the lowest non-zero bit in the binary representation of a.

2.1 Small Diameter

Theorem 4. *Given an n-node tree $T = (V, E)$, and a collection of paths $\mathcal{P} = \{P_1, \ldots, P_m\}$, there is a polynomial time algorithm which partitions $\mathcal{P}$ into $\ell = O(\log D)$ subsets $\mathcal{P}_1, \ldots, \mathcal{P}_\ell$, where D is the diameter of T, and provides a collection of edges F_i for each $\mathcal{P}_i$, such that each $P \in \mathcal{P}_i$ contains at least one and at most two edges of F_i.*

Proof. Let $\ell = \lceil \log(D + 1) \rceil$. Fix any vertex $r \in V$ as a root of the tree T. Let $H_j \subseteq E$ $(1 \le j \le D)$ be the set of edges which are at distance exactly j from r (the edges incident to r are in H_1). For each $1 \le i \le \ell$ we take

$$F_i = \bigcup_{j:1 \le j \le D, \texttt{POWER}_2(j)=i} H_j .$$

Note that as $1 \leq j \leq D$ we have $1 \leq \texttt{POWER}_2(j) \leq \lceil \log(D+1) \rceil = \ell$, and $\{F_1, \dots, F_\ell\}$ indeed is a partition of the set of edges (see Figure 1 for an illustration). Now we partition the set of requests $\mathcal{P}$ into subsets $\mathcal{P}_1, \dots, \mathcal{P}_\ell$. We put a request $P \in \mathcal{P}$ to the set $\mathcal{P}_i$ if P intersects F_i but does not intersect F_j for any $j > i$.

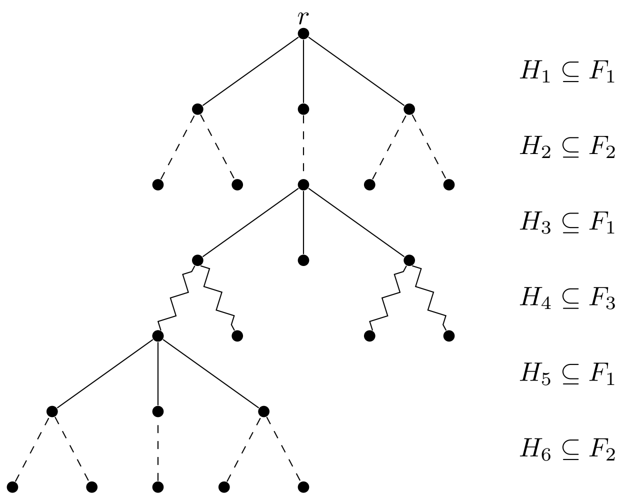

Fig. 1. Regular edges belong to F_1, dashed edges belong to F_2, whereas zigzagged edges belong to F_3.

As each request $P \in \mathcal{P}$ is a nonempty path, each request belongs to some set $\mathcal{P}_i$. By the definition, each request $P \in \mathcal{P}_i$ intersects the set F_i. It remains to prove that each $P \in \mathcal{P}_i$ contains at most two edges from the set F_i. For a request $P \in \mathcal{P}_i$ let j be an index, such that P contains an edge of H_j and $\texttt{POWER}_2(j) = i$ (note that such index j exists, since $P \in \mathcal{P}_i$). If P contains an edge from a set $H_{j'}$ where $1 \leq j' \leq D$, $\texttt{POWER}_2(j') = i$, $j' \neq j$, then P also contains an edge from some $H_{j''}$, where j'' is between j' and j and $\texttt{POWER}_2(j'') > \texttt{POWER}_2(j) = i$. But this is a contradiction with the assumption that P is in the set $\mathcal{P}_i$.

Finally, any path can contain at most two edges from any fixed H_j, since H_j is the set of edges equidistant from the root. Therefore each request $P \in \mathcal{P}_i$ contains at least one and at most two edges of F_i. $\square$

2.2 Large Diameter

Now we want to adapt ideas from the previous section to handle the case of large diameters, hence proving Theorem 1. To that aim, we exploit the following two technical lemmas. A subpath is *upward* if one endpoint is an ancestor of the other endpoint.

Lemma 5. *Given an n-node rooted tree T, there is a polynomial-time algorithm which partitions the edge set of T into upward paths, and groups such paths into $s = O(\log n)$ collections (called* levels*) $L_1, \ldots, L_s$ so that, for any path P of T the following two properties hold:*

1. **(separation)** *If P shares an edge with $P' \in L_i$ and $P'' \in L_j$, $i < j$, P must share an edge with some $P''' \in L_k$ for any $i < k < j$.*
2. **(intersection)** *P shares an edge with at most two paths in each level L_i, and if it shares an edge with two such paths P' and P'', then it must contain the topmost edges of the latter two paths.*

Proof. For each vertex v that is not a root nor a leaf, consider all edges of T that connect v with a child of v, and denote as e_v the edge that leads to a subtree with the largest number of vertices (breaking ties arbitrarily). Now, for each leaf w of T, consider a path P_w constructed as follows: start from w, traverse the tree towards the root r and stop when a vertex v is reached via an edge different than e_v. Let v_w be the topmost (i.e., the last, closest to the root r) vertex on the path P_w .

Since for each non-leaf and non-root vertex v, exactly one edge connecting v with its child is denoted e_v, each edge of T belongs to exactly one path P_w. For each vertex u of T, by $\text{depth}(u)$ we denote the number of different paths P_w that share an edge with the unique path connecting u with the root r. If a path P_w ends at a vertex $v_w \neq r$ and $v'v_w$ is the last edge of P_w, then, by the definition of the edge e_v, the subtree of T rooted at v_w has at least twice as many vertices than the subtree rooted at v'. We infer that for each vertex u we have $0 \leq \text{depth}(u) \leq \lceil \log n \rceil$; $\text{depth}(u) = 0$ iff $u = r$.

For each path P_w we denote $\text{depth}(P_w) = \text{depth}(v_w) + 1$; note that $1 \leq \text{depth}(P_w) \leq \lceil \log n \rceil + 1$. Let $s = \lceil \log n \rceil + 1$ and let $L_j = \{P_w : \text{depth}(P_w) = j\}$ for $1 \leq j \leq s$. Clearly the family $\{L_j : 1 \leq j \leq s\}$ can be constructed in polynomial (even linear) time and partitions the edges into levels, each level consisting of a set of upward paths (see Figure 2 for an illustration). Moreover, if paths P_{w_1} and P_{w_2} share a vertex, then $|\text{depth}(P_{w_1}) - \text{depth}(P_{w_2})| \leq 1$, which proves the separation property in the claim. It remains to prove the intersection property.

Consider any upward path P in the tree T. Clearly, the values $\text{depth}(v)$ for vertices v on the path P are ordered monotonously, with the lowest value in the topmost (closest to the root r) vertex of P. Therefore, for each layer L_j, the path P may contain edges of at most one path of L_j. Moreover, as all paths P_w are upward paths, if P contains an edge of P_w, then either P contains the topmost edge of P_w, or the topmost vertex of P lies on P_w, but is different from v_w. Since any path in T can be split into two upward paths, intersection property and the whole theorem follow. $\square$

The following lemma provides a way to partition requests.

Lemma 6. *Given a given rooted tree T and a set of requests $\mathcal{P}$, in polynomial time we can construct a collection of $\ell = O(\log \log n)$ families of paths*

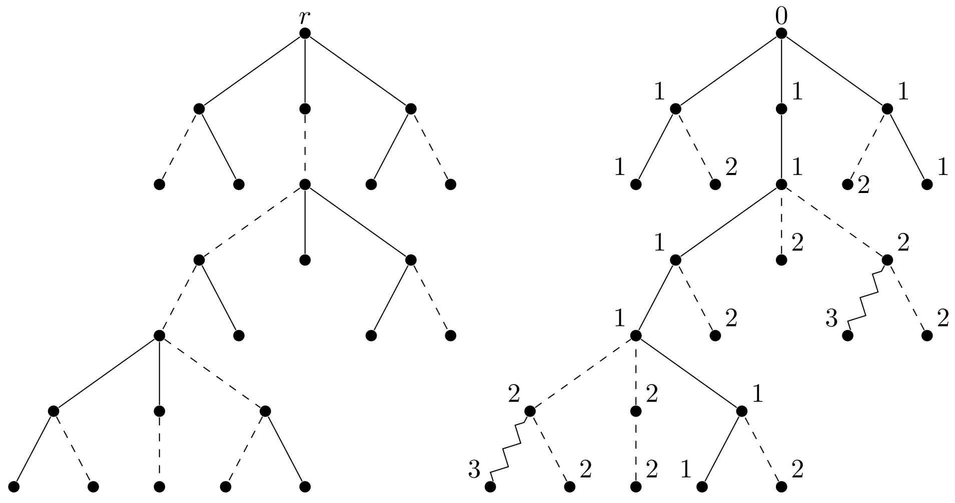

Fig. 2. In the left tree edges e_v are dashed. In the right tree nodes are labeled with their depth, regular edges belong to paths of level L_1, dashed edges belong to paths of level L_2, whereas zigzagged edges belong to paths of level L_3.

$G_1, G_2, \ldots, G_\ell$ in T (called groups*) and a partition $K_1, K_2, \ldots, K_\ell$ of $\mathcal{P}$, such that:*

1. *Each request $P \in K_i$ shares an edge with at at least one and at most two paths from the group G_i.*
2. *Each request $P \in K_i$ contains at most 4 edges incident to paths of G_i.*
3. *If a request $P \in K_i$ shares an edge with two paths $Q_1, Q_2 \in G_i$, then it contains the topmost edges of the latter two paths.*

Proof. First, invoke Lemma 5 on the tree T and obtain a partition into levels $L_1, \ldots, L_s$ with $s = O(\log n)$. Let $\ell = \lceil \log(s+1) \rceil = O(\log \log n)$ and, for $1 \le i \le \ell$, define G_i as follows:

$$G_i = \bigcup_{j : 1 \le j \le s, \mathtt{POWER}_2(j) = i} L_j \,.$$

Now partition the set of requests $\mathcal{P}$ into ℓ subsets K_i, for $1 \le i \le \ell$, in such a way that a request $P \in \mathcal{P}$ is assigned to the subset K_i iff P contains an edge of some path of G_i and does not contain any edge of any path of G_j for $j > i$. The paths of the groups G_i cover all edges from the tree T, so each request $P \in \mathcal{P}$ is included in some layer.

Observe, that due to the separation property ensured by Lemma 5, if a path $P \in \mathcal{P}$ belongs to K_i and P shares an edge with some path from L_j, $\mathtt{POWER}_2(j) = i$, then it does not share an edge with any path of level $L_{j'}$, $j' \neq j$, $\mathtt{POWER}_2(j') = i$. Indeed, otherwise P shares an edge of some path from $L_{j''}$,

$\texttt{POWER}_2(j'') > i$ and j'' is between j' and j. Thus, due to the intersection property of Lemma 5 the path P shares an edge with at most two paths of G_i. Analogously, if we split P into two upward paths P_1 and P_2, then each of them contains at most two edges incident to some path of G_i, and therefore P contains at most 4 edges incident to paths of G_i.

Finally, if a request $P \in K_i$ shares an edge with two paths $Q_1, Q_2 \in G_i$, then those two paths belong to the same level L_j and consequently by the intersection property of Lemma 5 the request P contains the topmost edges of both Q_1 and Q_2. □

We now have all the ingredients to prove Theorem 1.

Proof (of Theorem 1). First, invoke Lemma 6 to obtain groups $G_1, G_2, \ldots, G_\ell$ and a partition $K_1, K_2, \ldots, K_\ell$ of the set $\mathcal{P}$.

Next, consider each subset K_i independently. For a fixed subset of requests K_i we split the requests of K_i into two sets $\mathcal{P}_{2i-1}, \mathcal{P}_{2i}$:

– We put $P \in K_i$ into $\mathcal{P}_{2i-1}$ if it is entirely contained in some path Q of G_i;
– Otherwise we put P into $\mathcal{P}_{2i}$.

Note that for $\mathcal{P}_{2i-1}$ the levels give the desired partition $\mathcal{L} = \bigcup_{1 \le i \le \ell} G_i$ of the tree T such that each request in $\mathcal{P}_{2i-1}$ is entirely contained in some path of $\mathcal{L}$. Therefore, it remains to construct a set F_i such that each path of $\mathcal{P}_{2i}$ contains at least one and at most 6 edges of F_i.

As the set F_i we take all the topmost edges of paths in G_i as well as all edges in T incident to some path of G_i. Observe that, by Lemma 6, if a request $P \in \mathcal{P}_{2i}$ shares an edge with two paths Q_1 and Q_2 of G_i, then P contains the topmost edges of the latter two paths. Since no $P \in \mathcal{P}_{2i}$ is entirely contained in G_i, if P shares an edge with only one path of G_i, then it contains at least one incident edge to some path of G_i. Consequently, each path $P \in \mathcal{P}_{2i}$ contains at least one edge of F_i.

Finally, since each $P \in \mathcal{P}_{2i}$ contains at most 4 edges incident to paths of G_i and at most two topmost edges of paths of G_i, it contains at most 6 edges of F_i. □

3 Applications of the decomposition theorem

In this section we present two applications of our decomposition Theorem 1.

3.1 Uniform Tollbooth on Trees

Since we assume that the ratio between the smallest and greatest budget in an instance of Uniform Tollbooth on Trees is bounded by a constant and our goal is to obtain an $O(\log \log n)$-approximation, we can assume that all the budgets are equal to 1 and therefore we are going to work with the following definition.

Uniform Tollbooth on Trees (UTT)
Input: A tree $T = (V, E)$ with n vertices, and a set of paths $\mathcal{P} = \{P_1, \ldots, P_m\}$.
Task: Find a pricing function $p : E \to \mathbb{R}_+$ maximizing

$$\sum_{1 \le i \le m, p(P_i) \le 1} p(P_i)\,.$$

For an instance $\mathcal{I} = ((V, E), \mathcal{P})$ of UTT by $\mathrm{opt}(\mathcal{I})$ we denote the revenue obtained by an optimum solution. When the underlying tree does not change and we consider subsets $\mathcal{P}_i \subseteq \mathcal{P}$, then by $\mathrm{opt}(\mathcal{P}_i)$ we denote $\mathrm{opt}(((V, E), \mathcal{P}_i))$.

Theorem 7. *There is a polynomial time $O(\log\log n)$-approximation algorithm for* Uniform Tollbooth on Trees.

Proof. We use Theorem 1 and independently consider each of the subsets $\mathcal{P}_i$. We will obtain a constant factor approximation for each of the sets $\mathcal{P}_i$, which altogether gives an $O(\log\log n)$-approximation for UTT.

If a set $\mathcal{P}_i$ is of type (a), that is we are given a decomposition of the tree T into edge disjoint paths and each request of $\mathcal{P}_i$ is fully contained in one of the paths of the decomposition, then we can use a PTAS of Grandoni and Rothvoß [8] to obtain revenue at least $\mathrm{opt}(\mathcal{P}_i)/(1+\epsilon)$.

If a set $\mathcal{P}_i$ is of type (b), then we are additionally given a set of edges F_i, such that each each path of $\mathcal{P}_i$ contains at least one and at most 6 edges of F_i. Consequently, the pricing function defined as:

$$p(e) = \begin{cases} 1/6 & \text{if } e \in F_i \\ 0 & \text{otherwise} \end{cases}$$

gives at least $|\mathcal{P}_i|/6 \ge \mathrm{opt}(\mathcal{P}_i)/6$ revenue. □

3.2 Unique coverage on trees

Unique Coverage on Trees (UCT)
Input: A tree $T = (V, E)$ with n vertices, a set of paths $\mathcal{P} = \{P_1, \ldots, P_m\}$ and a profit function $p : \mathcal{P} \to \mathbb{R}_+$.
Task: Find a set of edges $X \subseteq E$, which maximizes the sum of profits of paths containing exactly one edge of X.

We start by presenting an exact polynomial-time algorithm for the UCT problem for the case when T is a path.

Lemma 8. *With an additional assumption that T is a path,* UCT *can be solved optimally in polynomial time.*

Proof. Let $\mathcal{I} = (T = (V, E), \mathcal{P}, p)$ be a UCT instance where T is a path and let $e_1, \ldots, e_{n-1}$ be the consecutive edges of T. Since each $P \in \mathcal{P}$ is a path, we can represent it as an interval $[a, b] \in P$, where $1 \le a \le b \le n-1$, e_a is the leftmost edge and e_b is the rightmost edge of P. We use a dynamic programming approach where we consider edges one by one and in a state we store the last two edges selected to the set X. Formally, for $1 \leq i < j < n$ we define $t[i,j]$ as a maximum profit of paths covered exactly once by a subset $X \subseteq E$, satisfying $X \cap \{e_i, \ldots, e_{n-1}\} = \{e_i, e_j\}$:

$$t[i,j] = \max_{X \subseteq E, X \cap \{e_i,\ldots,e_{n-1}\} = \{e_i,e_j\}} p(\{P \in \mathcal{P} : |P \cap X| = 1\}) \,.$$

Observe, that with this definition of entries of the 2-dimensional table t, the optimum profit is the maximum value of $t[i,j]$ over $1 \leq i < j < n$. It remains to show how to compute all the values $t[i,j]$ in polynomial time. We use the following recursive formula, where we either decide that $X = \{e_i, e_j\}$, or we iterate over the first edge in X to the left of e_i:

$$\begin{aligned} t[i,j] = \max(&p(\{P \in \mathcal{P} : |P \cap \{e_i, e_j\}| = 1\}), \\ &\max_{1 \leq k < i} (t[k,i] + p(A) - p(B_k))) \,. \end{aligned}$$

where $A \subseteq \mathcal{P}$ is the set of paths $[a,b] \in \mathcal{P}$ with $i < a \leq j \leq b$ and $B_k \subseteq \mathcal{P}$ is the set of paths $[a,b] \in \mathcal{P}$ with $k < a \leq i < j \leq b$. Note that when adding e_j to the set corresponding to $t[k,i]$ the paths in A start being covered uniquely, and the paths in B are being covered for the second time.

By the standard method of extending the table t with backlinks one can in polynomial time retrieve the set X corresponding to each of the values $t[i,j]$. $\square$

Theorem 9. *There is a polynomial time $O(\log\log n)$-approximation algorithm for the* UCT *problem.*

Proof. By using Theorem 1 for the tree T and the set of paths $\mathcal{P}$ it is enough to obtain a constant factor approximation for each set $\mathcal{P}_i$. Since for $\mathcal{P}_i$ with paths contained entirely in some path of the decomposition $\mathcal{L}$ we can use Lemma 8, it remains to handle type (b) of the set $\mathcal{P}_i$ given by Theorem 1.

Therefore we assume that each path of $\mathcal{P}_i$ contains at least one and at most 6 edges of the given set F_i. To the set X we independently take each edge of F_i with probability 1/6. Note, that with constant probability each path in $\mathcal{P}_i$ contains exactly one edge of X and therefore the expected profit given by X is a constant fraction of $p(\mathcal{P})$, which is a trivial upper bound on $\mathrm{opt}(\mathcal{P}_i)$. By the standard method of conditional expectation we can derandomize this procedure, obtaining a deterministic constant factor approximation for the second type of the set $\mathcal{P}_i$, which proves the theorem. $\square$

4 Conclusions

We have presented $O(\log\log n)$-approximation algorithms for Uniform Tollbooth on Trees. A natural question is whether a constant factor approximation is possible.

Moreover, obtaining a constant-, or even poly($\log\log n$)-approximation for Tollbooth on Trees with general budgets remains open. As a corollary of our techniques one can prove, that with a loss of a factor of $O(\log\log n)$ in the approximation ratio, one can assume that each request starts and ends in a leaf of the tree. We believe, that it is worthwhile to investigate the Tollbooth on Trees problem with general budgets, but with the additional assumption that the tree is a full binary tree and all the requests have their endpoints in the leaves of the tree.

Acknowledgements

We thank Guy Kortsarz for mentioning the tree case of the Unique Coverage problem as an interesting special case for which nothing better than the general $O(\log n)$ approximation was known.

Moreover we are thankful to anonymous referees for their helpful comments and remarks.

References

1. Maria-Florina Balcan and Avrim Blum. Approximation algorithms and online mechanisms for item pricing. In *ACM Conference on Electronic Commerce*, pages 29–35, 2006.
2. Nikhil Bansal, Amit Chakrabarti, Amir Epstein, and Baruch Schieber. A quasi-PTAS for unsplittable flow on line graphs. In *ACM Symposium on Theory of Computing (STOC)*, pages 721–729, 2006.
3. Patrick Briest and Piotr Krysta. Single-minded unlimited supply pricing on sparse instances. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1093–1102, 2006.
4. Erik D. Demaine, Uriel Feige, MohammadTaghi Hajiaghayi, and Mohammad R. Salavatipour. Combination can be hard: Approximability of the unique coverage problem. *SIAM Journal on Computing*, 38(4):1464–1483, 2008.
5. Khaled M. Elbassioni, Rajiv Raman, Saurabh Ray, and René Sitters. On profit-maximizing pricing for the highway and tollbooth problems. In *Symposium on Algorithmic Game Theory (SAGT)*, pages 275–286, 2009.
6. Khaled M. Elbassioni, René Sitters, and Yan Zhang. A quasi-ptas for profit-maximizing pricing on line graphs. In *European Symposium on Algorithms (ESA)*, pages 451–462, 2007.
7. Iftah Gamzu and Danny Segev. A sublogarithmic approximation for highway and tollbooth pricing. In *ICALP (1)*, pages 582–593, 2010.
8. Fabrizio Grandoni and Thomas Rothvoß. Pricing on paths: A ptas for the highway problem. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 675–684, 2011.
9. Venkatesan Guruswami, Jason D. Hartline, Anna R. Karlin, David Kempe, Claire Kenyon, and Frank McSherry. On profit-maximizing envy-free pricing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1164–1173, 2005.

10. Venkatesan Guruswami and Luca Trevisan. The complexity of making unique choices: Approximating 1-in- k sat. In *Workshop on Approximation Algorithms for Combinatorial Optimization Problems and Workshop on Randomization and Computation (APPROX-RANDOM)*, pages 99–110, 2005.
11. Jason D. Hartline and Vladlen Koltun. Near-optimal pricing in near-linear time. In *Workshop on Algorithms and Data Structures (WADS)*, pages 422–431, 2005.
12. Rohit Khandekar, Tracy Kimbrel, Konstantin Makarychev, and Maxim Sviridenko. On hardness of pricing items for single-minded bidders. In *Workshop on Approximation Algorithms for Combinatorial Optimization Problems and Workshop on Randomization and Computation (APPROX-RANDOM)*, pages 202–216, 2009.
13. Preyas Popat and Yi Wu. On the hardness of pricing loss-leaders. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 735–749, 2012.