Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 223 (2023) 148–156

Procedia Computer Science

www.elsevier.com/locate/procedia

XII Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS 2023)

Min-max coverage problems on tree-like metrics

Eric Aaron[a], Úrsula Hébert-Johnson[b], Danny Krizanc[c], Daniel Lokshtanov[b]

[a]*Colby College, Waterville, ME, 04901, USA*
[b]*University of California, Santa Barbara, 93106, USA*
[c]*Wesleyan University, Middletown, CT, 06457, USA*

Abstract

We consider a number of min-max coverage problems. In each problem, the input is an unweighted graph G and an integer k, and possibly some additional information, such as a root vertex r. In the Min-Max Path Cover problem, the task is to cover all vertices of the graph by k walks, minimizing the length of the longest walk. The variant of Min-Max Path Cover in which all walks start and end at the same prescribed root vertex r is called the k-Traveling Salesmen Problem. In the Min-Max Tree Cover problem, the task is to cover all vertices of the graph by k trees, minimizing the size (number of edges) of the largest tree. In the rooted version, Min-Max k-Rooted Tree Cover, the input also contains k roots $r_1, \ldots, r_k$, and the ith tree must contain the root r_i. These four problems are all known to be APX-hard and to admit a constant-factor approximation. In this paper, we initiate the systematic study of these problems on trees and, more generally, on graphs of constant treewidth. As opposed to most graph problems, all four of the above coverage problems remain NP-hard even when G is a tree. We obtain an $n^{O(k)}$-time exact algorithm for all four problems on graphs of bounded treewidth. Our main contribution is a quasi-polynomial-time approximation scheme (QPTAS) for the k-Traveling Salesmen Problem, Min-Max Path Cover, and Min-Max Tree Cover on graphs of bounded treewidth.

Keywords: Approximation algorithms; Parameterized algorithms; Min-max coverage; Treewidth; Vehicle routing

1. Introduction

We consider a number of min-max coverage problems on graphs of bounded treewidth. The task is to find k objects that together cover the entire vertex set of the graph, so that the largest one of the objects is as small as possible. In general, the input graph G can have positive edge lengths $d\colon E(G) \to \mathbb{R}^+$ or be unweighted — in this paper, our results are concerned with unweighted graphs, i.e., those where $d(e) = 1$ for every edge e. By varying the types of objects used and by specifying additional information in the input, we obtain different problem formulations. For a concrete example, in Min-Max Path Cover the task is to cover $V(G)$ by k walks such that the length of the longest walk (number of edges) is minimized. A walk is a sequence of vertices such that each pair of consecutive vertices are adjacent. In Min-Max Tree Cover, the task is to cover $V(G)$ by k trees, minimizing the number of edges in the largest tree.

10.1016/j.procs.2023.08.224

Min-Max Path Cover and Min-Max Tree Cover draw their motivation from applications in robotics, operations research, and motion planning [1, 2, 3, 4]. Each walk in Min-Max Path Cover can correspond to the movement of a robot or vehicle, while the vertices represent locations that need to be visited. If the robots all start simultaneously and move at the same speed, then the maximum length of a walk corresponds to the time when the last location is visited. In these applications, it is likely that the planner does not have full control over the starting/ending points of the robots. Thus various *rooted* variants of these problems have been studied (see [5] and references within). In this work, we consider the k-Traveling Salesmen Problem (k-TSP), which is the version of Min-Max Path Cover where every walk must start and end at the same prescribed root vertex r. We also consider the Min-Max k-Rooted Tree Cover problem, where the input contains k root vertices $r_1, r_2, \ldots, r_k$ and the ith tree must contain r_i.

All of the above problems depend only on the shortest-path metric of G. In other words, the optimal solution is the same in G as it is in the complete graph with vertex set $V(G)$ in which each edge uv has weight $d_G(u, v)$, i.e., the length (sum of the weights) of the shortest path between u and v in G. For this reason, several works (e.g., [6, 5]) assume that G is a *metric* — a complete graph whose edge lengths satisfy the triangle inequality. In such a graph, we can skip over any vertex that has already been visited. This is likely the reason behind the slightly confusing name for Min-Max Path Cover, considering that we cover the graph by walks rather than paths.

The above problems are all known to be NP-hard [5], even when the number of parts k is constant. At the same time, for all of them, constant-factor approximations are known [6, 7]. Indeed, several of these problems have seen a string of polynomial-time approximation algorithms, achieving better and better approximation ratios [8, 7, 6, 5]. For example, [8] and [6] each give a 4-approximation algorithm for Min-Max Tree Cover (with positive integer edge weights), which are both improved upon by the 3-approximation presented in [7]. However, assuming P ≠ NP, each string of improvements has to converge, since these problems are all known to be APX-hard, even on unweighted graphs [9]. For instance, in the case of Min-Max Tree Cover, the APX-hardness lower bound is $\frac{3}{2}$. This motivates the study of min-max coverage problems on classes of graphs for which these problems become more tractable.

1.1. Our results

In this work, we initiate the systematic study of min-max coverage problems on unweighted trees and unweighted graphs of constant treewidth. Unlike most graph problems, which become trivially solvable in polynomial time on trees, all of the above problems remain NP-hard on trees. (This can be proved via a reduction from 3-Partition for each problem — see the full version of the paper for details.) Therefore, we switch our focus to parameterized [10] and approximation algorithms [11] for these problems on trees and graphs of bounded treewidth. Our first positive result is an $n^{4k+1}\omega^{O(k\omega)}$-time algorithm for a class of min-max coverage problems on graphs of treewidth ω. In particular, this result implies a $2^{O(k)}n^{4k+1}$-time algorithm on trees for all of the above problems: k-TSP, Min-Max Path Cover, Min-Max Tree Cover, and Min-Max k-Rooted Tree Cover. Furthermore, if we define Map Visitation as the variant of Min-Max Path Cover where the input contains k root vertices $r_1, r_2, \ldots, r_k$ and the ith walk must start at r_i (but can end anywhere), then this algorithm patches an error in a claimed polynomial-time algorithm for Map Visitation on trees with $k = 2$ [12]. In the full version of the paper, we also show two corresponding lower bounds: (1) the above class of min-max coverage problems on trees is W[1]-hard[1] parameterized by k; (2) assuming ETH, there is no $f(k)n^{o(k/\log k)}$-time algorithm for this class of problems on trees.

Our main technical contribution is a quasi-polynomial-time approximation scheme (QPTAS) for a class of min-max coverage problems on graphs of treewidth ω. The algorithm obtains a $(1 + \epsilon)$-approximate solution in time $n \cdot k^{(\log n)^2\omega^{O(\omega)}/\varepsilon}$. This algorithm still works for most of the problems discussed so far (namely k-TSP, Min-Max Path Cover, and Min-Max Tree Cover), but it is slightly less general than our exact algorithm. In particular, the QPTAS fails for *asymmetric* coverage problems such as Min-Max k-Rooted Tree Cover. "Asymmetric" refers to the fact that the different solution trees are no longer interchangeable, since distinct trees are allowed to have distinct roots. In the full version of the paper, we show that the more general class of problems which allows asymmetry is APX-hard even on unweighted trees.

[1] This implies that there is no FPT algorithm (i.e., one with running time $f(k)n^{O(1)}$) for this class of problems unless FPT = W[1].

1.2. Our methods

The $n^{4k+1}\omega^{O(k\omega)}$-time exact algorithm on graphs of treewidth ω is a fairly standard dynamic-programming algorithm. We consider a rooted tree decomposition, and for every node t of the decomposition tree T we construct a family of partial solutions for G_t, the subgraph of G induced by the union of all bags in the subtree of T rooted at t. The crux of the algorithm is to group partial solutions into $n^k\omega^{O(k\omega)}$ equivalence classes, such that any two partial solutions in the same class are interchangeable (i.e., if one can be completed to an optimal solution then so can the other). Thus we only keep at most one solution from each equivalence class at each node t. Let X_t be the bag at node t. The grouping of partial solutions into equivalence classes is based on how many edges of G_t are visited by each walk/tree (the factor n^k in the number of equivalence classes comes from this), and on which vertices of X_t are in the "same component" of the walk/tree.

While the exact algorithm is fairly standard, it serves as a warm-up for the QPTAS. The QPTAS is built on a new twist to the technique of Lampis [13], who showed that for many problems that admit n^ω-time algorithms when parameterized by treewidth ω, if that running time comes from ω numbers ranging between $1, \ldots, n$, then one can often obtain $(1+\epsilon)$-approximation algorithms with running time $f(\epsilon, \omega)n^{O(1)}$. On the one hand, this technique seems applicable to our setting since G has bounded treewidth and the hardness appears to come from large numbers. On the other hand, this technique appears *not* to be applicable because ω is not the parameter in the exponent of n. In fact, the problem is already NP-hard for $\omega = 1$.

Following Lampis [13], we store approximate values of the costs (numbers of edges visited in G_t) in a partial solution, in such a way that the error is at most $1+\varepsilon$. As is, this gives a running time of about $(\omega^{O(\omega)} \cdot (\log n)^2)^{O(k)}$, when ε is fixed. To obtain a QPTAS, the crucial observation is that now, the number of essentially different ways (types) that each of the k walks/trees can intersect with G_t is bounded by $(\log n)^2\omega^{O(\omega)}$. For *symmetric* coverage problems with interchangeable parts (walks/trees), it is sufficient to store how many parts there are of each type. Since there are $(\log n)^2\omega^{O(\omega)}$ types and at most k parts of each type, the number of equivalence classes (which dominates the running time) drops down to $k^{(\log n)^2\omega^{O(\omega)}}$.

To the best of our knowledge, this is the first use of the method of Lampis [13] to obtain an approximation algorithm that runs in (quasi) polynomial time when the relevant parameter is unbounded. Typically, this method transforms an exact XP algorithm into an FPT approximation algorithm that runs in exponential time when the parameter is unbounded. However, we transform an exact XP algorithm (parameterized by k) into an approximation algorithm that runs in quasi-polynomial time even when $k = n$. We believe this proof of concept will lead to further applications of this method in the design of approximation algorithms.

2. Exact algorithm for Min-Max Coverage on graphs of bounded treewidth

We define a general problem framework called Min-Max Coverage, encompassing k-TSP, Min-Max Path Cover, Min-Max Tree Cover, and Min-Max k-Rooted Tree Cover. In this setting, we wish to cover the vertices of the input graph G by k walks/trees, which we call *sections*. This framework will allow us to exploit the strong similarities between these four problems.

The input to Min-Max Coverage is a graph G, a positive integer k (the number of sections), an identifier $\ell \in$ {"Path Cover", "Tree Cover"} that specifies which type of problem we are solving, a list of sets $S_1, \ldots, S_k \subseteq V(G)$ of allowable starting points, and a list of sets $Z_1, \ldots, Z_k \subseteq V(G)$ of allowable ending points. (The starting/ending point terminology here stems from Path Cover. For Tree Cover, we can think of these points as roots.) Naturally, to solve k-TSP or Min-Max Path Cover we set $\ell =$ "Path Cover", and to solve either of the Tree Cover problems, we set $\ell =$ "Tree Cover".

For k-TSP, we set $S_i = Z_i = \{r\}$ for all $i \in [k]$. For Min-Max Path Cover and for Min-Max Tree Cover, we set $S_i = Z_i = V(G)$ for all $i \in [k]$. For Min-Max k-Rooted Tree Cover, we set $S_i = Z_i = \{r_i\}$ for each $i \in [k]$. This framework also encompasses more general problems than the four above, such as the version of Path Cover in which each walk is given an arbitrary set of allowable starting points and allowable ending points. We can formulate similar variations of Tree Cover as well.

Feasible solutions can be expressed as follows. A *function-tuple* is a k-tuple of functions $f = (f_1, \ldots, f_k)$, where $f_i \colon E(G) \to \{0, 1, 2\}$ for all $i \in [k]$. For such a function f_i, we define G_{f_i} to be the subgraph of G whose edge set is

$\{e \in E(G) : f_i(e) \geq 1\}$ and whose vertex set consists of just the endpoints of those edges. In addition, for a function f_i and a vertex $v \in V(G)$, we define $\deg_{f_i}(v) \coloneqq \sum_{e \text{ inc with } v} f_i(e)$. Here the sum is over all edges incident with v.

Definition 1. A *feasible solution* to Min-Max Coverage is a function-tuple $f = (f_1, \ldots, f_k)$, along with a starting point $s_i \in S_i$ and an ending point $z_i \in Z_i$ for all $i \in [k]$, that satisfy the following:

1. The graph G_{f_i} is connected for all $i \in [k]$.
2. For all $v \in V(G) \setminus \{s_1, \ldots, s_k, z_1, \ldots, z_k\}$, $\deg_{f_i}(v) \geq 1$ for some $i \in [k]$, i.e., every vertex is covered.
3. For all $i \in [k]$, one of the following is true:
   - $\deg_{f_i}(s_i) \geq 1$ and $\deg_{f_i}(z_i) \geq 1$,
   - $s_i = z_i$, and $f_i(e) = 0$ for all $e \in E(G)$.
4. If $\ell =$ "Path Cover", then for all $i \in [k]$, the following statements hold:
   - $\deg_{f_i}(v)$ is even for all $v \in V(G) \setminus \{s_i, z_i\}$,
   - if $s_i = z_i$, then $\deg_{f_i}(s_i)$ and $\deg_{f_i}(z_i)$ are both even,
   - if $s_i \neq z_i$, then $\deg_{f_i}(s_i)$ and $\deg_{f_i}(z_i)$ are both odd.

The third requirement checks that each section respects the starting and ending points. The fourth ensures that for each section in the covering, all degrees are even if that section is a closed walk, and all vertices except for s_i and z_i have even degree if $s_i \neq z_i$. Finally, the objective is to minimize $\max_{i \in [k]} \sum_{e \in E(G)} f_i(e)$, i.e., minimize the size of the largest section in the covering. For a feasible solution f, the *cost* of f refers to the quantity $\max_{i \in [k]} \sum_{e \in E(G)} f_i(e)$.

Theorem 1. *Let G be an n-vertex graph, given with its tree decomposition of width ω. Then* Min-Max Coverage *on G with k sections can be solved in time $n^{4k+1}\omega^{O(k\omega)}$.*

This algorithm will be precisely what we need to solve the four problems of interest. Specifically, for k-TSP or Min-Max Path Cover, each ith walk can be represented by a function $f_i \colon E(G) \to \mathbb{N}_{\geq 0}$. For each $e \in E(G)$, $f_i(e)$ specifies how many times the edge e is visited by the ith walk. In fact, for both of these problems, it is sufficient to only consider feasible solutions that visit each edge at most twice. Indeed, given a feasible solution that visits some edge more than twice, by Euler's theorem, we can convert this into a solution of the desired form with at most the same cost in the following way: for each $e \in E(G)$, if $f_i(e)$ is odd (resp. even) and nonzero, then set $f_i(e) = 1$ (resp. $f_i(e) = 2$). In both Tree Cover problems, each ith tree can be represented by a function $f_i \colon E(G) \to \{0, 1\}$. We let $f_i(e) = 1$ if e belongs to the ith tree, and $f_i(e) = 0$ otherwise. We can also solve this problem in the framework that allows f_i to map to $\{0, 1, 2\}$ since we can assume each f_i in the optimal solution maps to $\{0, 1\}$. In addition, we can assume the optimal solution is acyclic since any feasible solution containing a cycle can be improved to obtain an acyclic solution. Therefore, Theorem 1 immediately implies the following corollary:

Corollary 2. *Let G be an n-vertex graph, given with its tree decomposition of width ω. Then* k-TSP, Min-Max Path Cover, Min-Max Tree Cover, *and* Min-Max k-Rooted Tree Cover *can all be solved on G in time $n^{4k+1}\omega^{O(k\omega)}$. Here k is the number of walks/trees in the covering.*

To prove Theorem 1, suppose we have an algorithm for the special case in which each set S_i, Z_i contains a single vertex. Given such an algorithm, we can guess the starting and ending point for each section to solve the problem with arbitrary sets S_i, Z_i, incurring only an additional factor of n^{2k} in the running time. Therefore, it is sufficient to prove the following:

Theorem 3. *Let G be an n-vertex graph, given with its tree decomposition of width ω. Then* Min-Max Coverage *with k sections on G with singleton starting point sets $\{s_1\}, \ldots, \{s_k\}$ and ending point sets $\{z_1\}, \ldots, \{z_k\}$ can be solved in time $n^{2k+1}\omega^{O(k\omega)}$.*

Exact algorithm overview. At each node in the tree decomposition, we store a collection of "signatures" representing partial solutions. The signature of a partial solution includes the cost (so far) for each section, as well as information

that captures in what ways it can be feasibly extended to the remainder of G. The algorithm moves upward from the leaves to the root, so that in the end the set of signatures at the root allows us to solve the coverage problem for G.

We assume we are given a nice tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of G with introduce vertex nodes, introduce edge nodes, forget vertex nodes, and join nodes (see [10] for details). In particular, we assume our nice tree decomposition has the following properties: every edge in $E(G)$ is introduced exactly once, and every vertex in $V(G)$ is forgotten exactly once, although a vertex may be introduced multiple times. For $t \in V(T)$, let $E_t \subseteq E(G)$ be the set of edges introduced in the subtree of T rooted at t, and let $Y_t \subseteq V(G)$ be the set of vertices forgotten in the subtree of T rooted at t.

A *partial solution with respect to a node* $t \in V(T)$ is a k-tuple of functions $f = (f_1, \ldots, f_k)$, where $f_i \colon E_t \to \{0, 1, 2\}$ for all $i \in [k]$. We no longer need to include s_i and z_i in a (partial or feasible) solution since those are now given in the input. For a partial solution $f = (f_1, \ldots, f_k)$ with respect to t, and for $v \in V(G)$, we define the *degree* of each f_i at v as follows: $\deg_{f_i}(v) \coloneqq \sum_{e \in E_t,\, e \text{ inc with } v} f_i(e)$. This generalizes the definition of $\deg_{f_i}(v)$ above, since at the root r of the tree decomposition, we have $E_r = E(G)$. For a partial solution $f = (f_1, \ldots, f_k)$ with respect to t, for $i \in [k]$, we define $G_{f_i}^{(t)}$ to be the subgraph of G whose edge set is $\{(u, v) \in E(G) : f_i(u, v) \geq 1 \text{ and } u, v \in Y_t\}$ and whose vertex set consists of the endpoints of those edges. Intuitively, $G_{f_i}^{(t)}$ consists of the components of G_{f_i} that have been forgotten.

Definition 2. Let $f = (f_1, \ldots, f_k)$ be a partial solution with respect to t. We say that f is *feasible* if it satisfies the following requirements for every $v \in Y_t$:

1. The graph $G_{f_i}^{(t)}$ is connected for all $i \in [k]$.
2. If $v \notin \{s_1, \ldots, s_k, z_1, \ldots, z_k\}$, then $\deg_{f_i}(v) \geq 1$ for some $i \in [k]$.
3. For all $i \in [k]$ such that $v \in \{s_i, z_i\}$ and $\deg_{f_i}(v) = 0$, we have $s_i = z_i$, and $f_i(e) = 0$ for all $e \in E_t$.
4. If $\ell =$ "Path Cover", then for all $i \in [k]$, the following holds:
   - if $v \notin \{s_i, z_i\}$ or $v = s_i = z_i$, then $\deg_{f_i}(v)$ is even,
   - if $v \in \{s_i, z_i\}$ and $s_i \neq z_i$, then $\deg_{f_i}(v)$ is odd.

Intuitively, a feasible partial solution is one in which the forgotten vertices have been covered in a "feasible" way. Note that a partial solution, or even a feasible partial solution, does not necessarily extend to some feasible solution for G. However, at the root of the tree decomposition, feasible partial solutions with respect to the root are exactly the same as feasible solutions for G.

Let $f = (f_1, \ldots, f_k)$ be a partial solution with respect to t. The *weight* of f_i on E_t is defined as $w_{f_i}(E_t) \coloneqq \sum_{e \in E_t} f_i(e)$. In words, this is the cost of section i, restricted to edges in E_t. We define $E_t^{(0)}$ to be the set of edges in E_t having neither endpoint in the bag X_t, and we define $E_{t,i}^{+} \coloneqq \{e \in E_t : f_i(e) \geq 1\}$, i.e., this is the subset of E_t consisting of edges that section i covers at least once. In the following, the phrases in normal text are the formal definition, and the phrases in italics are intuition.

Definition 3. Suppose $f = (f_1, \ldots, f_k)$ is a partial solution with respect to some $t \in V(T)$. The *signature* of f consists of five components:

1. A k-tuple of costs $(c_1, \ldots, c_k)$, where $c_i = w_{f_i}(E_t)$ for each $i \in [k]$, *i.e., c_i is the cost of section i on the edges introduced thus far*.
2. For each $v \in X_t$, $i \in [k]$, we have a boolean $b_{v,i}$, where $b_{v,i} = 1$ if and only if $\deg_{f_i}(v) \geq 1$.
3. For each $v \in X_t$, $i \in [k]$, we have a boolean $p_{v,i}$, where $p_{v,i} \equiv \deg_{f_i}(v) \pmod 2$.
4. For each $i \in [k]$, we have a boolean q_i, where $q_i = 1$ if and only if either one of the following holds:
   - There exists an edge $e \in E_t^{(0)}$ such that $f_i(e) \geq 1$ and such that there is no path in G from an endpoint of e to a vertex in X_t using only edges from $E_{t,i}^{+}$, *i.e., section i has been forgotten and is disconnected from X_t*.
   - We have $s_i = z_i \in Y_t$, and $\deg_{f_i}(s_i) = 0$, *i.e., section i is the single vertex $s_i = z_i$, and this vertex has been forgotten*.

5. For each $i \in [k]$, we have a partition $\mathcal{P}_i$ of X_t; for each pair of vertices $u, v \in X_t$, u and v belong to the same part in $\mathcal{P}_i$ if and only if there exists a path from u to v in G using only edges from $E^+_{t,i}$, *i.e., u and v belong to the same part if and only if there is a path from u to v using only (introduced) edges from section i; in particular, any vertex not covered by section i belongs to a singleton part in* $\mathcal{P}_i$.

For a partial solution f whose k-tuple of costs in its signature is $(c_1, \ldots, c_k)$, the *cost* of f is $\max_{i\in[k]} c_i$. At each node $t \in T$, we store the set of all signatures of feasible partial solutions with respect to t that have cost at most $2n-2$. We denote this set by $c[t]$. We can jettison any solutions of cost greater than $2n-2$ since whenever a feasible solution exists, there must be a feasible solution of cost at most $2n-2$ (in particular, one can be obtained from a spanning forest of G). This means there are at most $(2n)^k$ possible k-tuples of costs for a signature at a given node. Overall, one can show that there are at most $n^k\omega^{O(k\omega)}$ distinct possible signatures at each node, which leads to the desired running time. The full details of the exact algorithm (including how the signatures are computed at each type of node in the tree decomposition), as well as the running time analysis and the proof of correctness, can all be found in the full version of the paper. What follows is a brief overview of the proof of correctness.

Soundness and completeness. For a feasible solution $f = (f_1, \ldots, f_k)$ for G and a node $t \in V(T)$, we say f *restricted to* E_t to refer to the partial solution $(f_1|_{E_t}, \ldots, f_k|_{E_t})$, and we denote this by $f|_{E_t}$. For $t \in V(T)$, we say that $c[t]$ contains a *complete* set of signatures for t if the following is true: for every feasible solution f for G of cost at most $2n-2$, $c[t]$ contains the signature of $f|_{E_t}$ with respect to t. For the root r of T, we say that $c[r]$ is a *sound* set of signatures for r if the "converse" is true: for every signature $\Pi \in c[r]$, there exists a feasible solution f such that Π is the signature of f with respect to r. In the full version of the paper, we prove the following two lemmas:

Lemma 4. *For all $t \in V(T)$, $c[t]$ contains a complete set of signatures for t.*

Lemma 5. *At the root r of T, $c[r]$ is a sound set of signatures for r.*

Although we only prove soundness for the root of the tree decomposition, the stronger statement mentioned above in fact holds: at every node t, $c[t]$ is exactly the set of all signatures of feasible partial solutions with respect to t of cost at most $2n-2$. We mention this because it may provide some intuition for which signatures are stored at each node, and which are filtered out.

3. QPTAS for Symmetric Min-Max Coverage on graphs of bounded treewidth

In our exact algorithm, the $n^{O(k)}$ bottleneck in the running time comes from the fact that there are $(2n)^k$ possibilities for the k-tuple of costs in each signature. We labeled the sections as $1, 2, \ldots, k$, and the signature of a partial solution included some information about each section i, including its cost c_i. However, we could have taken a slightly different approach. We could have defined the "type" of a section to be its cost c_i along with the booleans and partition for that particular section, and then we could have defined the signature of a partial solution to keep track of how many sections there are of each type. This would have saved space in the dynamic-programming table by a factor of up to $k!$ in symmetric instances, which is quite small considering the running time of the exact algorithm, so we chose to label the distinct sections for a slightly cleaner presentation. However, in our approximation algorithm, we will round up each cost c_i to an approximate value in such a way that there are only $O(\frac{(\log n)^2}{\varepsilon})$ distinct possible approximate costs, rather than $O(n)$ possibilities. Therefore, taking the latter approach with types will lead to a dramatic time savings.

To ensure that we can group together sections of the same type, we restrict our attention to *symmetric* coverage problems where the sections are interchangeable according to their allowable starting and ending points. Our QPTAS solves the problem of Symmetric Min-Max Coverage, which is defined to include all instances of Min-Max Coverage in which the starting and ending point sets are symmetric, i.e., we have sets S and Z such that $S_i = S$ and $Z_i = Z$ for all $i \in [k]$.

Theorem 6. *Let G be an n-vertex graph, given with its tree decomposition of width ω. Given any $\varepsilon > 0$, for the problem of* Symmetric Min-Max Coverage *on G with k sections, one can compute a $(1+\varepsilon)$-approximate solution in time $n \cdot k^{(\log n)^2\omega^{O(\omega)}/\varepsilon}$.*

In particular, Theorem 6 immediately implies the following corollary:

Corollary 7. *Let G be an n-vertex graph, given with its tree decomposition of width ω. Given any $\varepsilon > 0$, for the problems k-TSP,* Min-Max Path Cover, *and* Min-Max Tree Cover *on G, one can compute a $(1+\varepsilon)$-approximate solution in time $n \cdot k^{(\log n)^2 \omega^{O(\omega)}/\varepsilon}$. Here k is the number of walks/trees in the covering.*

QPTAS algorithm overview. At each node in the tree decomposition, we store a collection of "approximate signatures" representing partial solutions, each of which keeps track of the number of sections of each "approximate type." The approximate type of a section contains an approximation of its cost c_i, as well as its booleans and partition. This allows us to handle sections having the same approximate type collectively, rather than storing information about each section individually. This idea is based on the technique of Lampis [13] in which costs are computed using approximate addition trees.

We first modify the given tree decomposition so that it has depth $d = O(\omega \log n)$. This will allow us to keep the magnitude of the round-off errors under control as we proceed up the decomposition tree. To see why this leads to a $(1+\varepsilon)$-approximate solution, suppose we have a partial solution whose k-tuple of costs is $(c_1, \ldots, c_k)$. Let $\delta = \frac{\varepsilon}{2d}$. We round each cost c_i up to the nearest power of $1+\delta$ so that in the resulting k-tuple $(\widetilde{c_1}, \ldots, \widetilde{c_k})$, there are now only about $\log_{1+\delta}(2n)$ possible "buckets" for each $\widetilde{c_i}$. This rounding procedure introduces a multiplicative error of at most $1+\delta$ at each level (i.e., distance from the root) of the tree decomposition. Since we have chosen δ so that $(1+\delta)^d \leq 1+\varepsilon$, this gives a multiplicative error of at most $1+\varepsilon$ at the root.

Running time. The running time hinges on the number of distinct possible approximate costs. One can show that this value is essentially $\log_{(1+\delta)}(2n) \leq \frac{4(\log n)^2}{\varepsilon}$. Hence there are $k^{(\log n)^2 \omega^{O(\omega)}/\varepsilon}$ possible approximate signatures at each node, since each signature stores the number of sections of each approximate type: clearly there are at most k sections of each type, and one can show that there are at most $\frac{(\log n)^2}{\varepsilon} \cdot \omega^{O(\omega)}$ possible types. This gives us the claimed running time. The factor of n appears in the running time since we can assume there are $O(n)$ nodes in the given tree decomposition [10].

3.1. QPTAS setup, definitions, and algorithm

At the start of the QPTAS, we use the following result (Lemma 2.2 in [14]) to obtain a tree decomposition of depth $O(\log n)$ for G. This result is originally a parallel algorithm, but here we use its sequential form.

Proposition 8 (Bodlaender and Hagerup [14])**.** *There is an algorithm that, given a tree decomposition for G of width ω, finds a rooted binary-tree decomposition of G of depth $O(\log n)$ and width at most $3\omega + 2$ in $O(\omega n)$ time.*

From this tree decomposition of depth $O(\log n)$, we can now obtain a nice tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of G of depth $d = O(\omega \log n)$ and width at most $3\omega + 2$ in time $O(\omega^2 n)$ (Lemma 7.4 in [10]). This nice tree decomposition $\mathcal{T}$ has introduce vertex nodes, introduce edge nodes, forget vertex nodes, and join nodes. This is the tree decomposition that will be used from now on.

Next, we describe our notions of approximate costs, types, and signatures. As mentioned above, let $\delta = \frac{\varepsilon}{2d}$. For $t \in V(T)$, let $V_t \subseteq V(G)$ be the set of vertices introduced in the subtree of T rooted at t. Recall that $E_t \subseteq E(G)$ is the set of edges introduced in this subtree, and recall that $Y_t \subseteq V(G)$ is the set of vertices forgotten in this subtree.

Since S and Z can now contain multiple vertices, we must extend the definition of a partial solution slightly. From now on, a *partial solution with respect to t* consists of a k-tuple of functions $(f_1, \ldots, f_k)$ as before, where $f_i \colon E_t \to \{0, 1, 2\}$ for all $i \in [k]$, and in addition, we include the values of the starting and ending points $s_1, \ldots, s_k \in (S \cap X_t) \cup \{\text{NULL}, \text{DEFINED}\}$ and $z_1, \ldots, z_k \in (Z \cap X_t) \cup \{\text{NULL}, \text{DEFINED}\}$. The NULL values allow for the possibility of later extending some sections to start or end at a vertex outside of V_t. A value of DEFINED indicates that this staring or ending point belongs to $Y_t = V_t \setminus X_t$ (in this case we do not need to store the exact value).

Definition 4. Suppose $f = ((f_1, \ldots, f_k), (s_1, \ldots, s_k), (z_1, \ldots, z_k))$ is a partial solution with respect to some $t \in V(T)$. Fix $i \in [k]$. A *semi-approximate type* of section i for the partial solution f consists of six components:

1. A nonnegative integer B that satisfies $w_{f_i}(E_t) \leq B \leq (1+\delta)^{d'} w_{f_i}(E_t)$, where d' is the distance from t to its lowest leaf descendant in the tree decomposition. *This is an approximation of $w_{f_i}(E_t)$, i.e., this tells us which bucket section i falls into according to its cost on edges introduced thus far.*
2. For each $v \in X_t$, we have a boolean b_v, where $b_v = 1$ if and only if $\deg_{f_i}(v) \geq 1$.

3. For each $v \in X_t$, we have a boolean p_v, where $p_v \equiv \deg_{f_i}(v) \pmod 2$.
4. A boolean q, where $q = 1$ if and only if either one of the following holds:

   - There exists an edge $e \in E_t^{(0)}$ such that $f_i(e) \geq 1$ and such that there is no path in G from an endpoint of e to a vertex in X_t using only edges from $E_{t,i}^+$.
   - We have $s_i = z_i \in Y_t$ and $\deg_{f_i}(s_i) = 0$.

5. A partition $\mathcal{P}$ of X_t; for each pair of vertices $u, v \in X_t$, u and v belong to the same part in $\mathcal{P}$ if and only if there exists a path from u to v in G using only edges from $E_{t,i}^+$.
6. A starting point s and an ending point z, where $s = s_i$ and $z = z_i$.

A section i in a partial solution f can have various possible semi-approximate types since there may be various values of B that satisfy $w_{f_i}(E_t) \leq B \leq (1+\delta)^{d'} w_{f_i}(E_t)$. However, for each section in f, there is one value of B that is computed by our algorithm, at each node t. The semi-approximate type whose first component is that particular value of B is called the *approximate type* of section i in f.

Definition 5. Suppose $f = ((f_1, \ldots, f_k), (s_1, \ldots, s_k), (z_1, \ldots, z_k))$ is a partial solution with respect to some $t \in V(T)$. The *approximate signature* of f is the multiset $\{\alpha : \alpha \text{ is the approximate type of some section in } f\}$. The multiplicity of each approximate type α in this multiset is the number of sections having approximate type α in f. Thus the sum of the multiplicities is k.

As mentioned above, at each node in the tree decomposition, we store a set of approximate signatures, denoted by $c[t]$, representing partial solutions with respect to t. We compute $c[t]$ bottom-up, from the leaves to the root. The full details of the algorithm, as well as the running time analysis and the proof of correctness, can all be found in the full version of the paper.

4. Conclusion

Our main result is a QPTAS for Symmetric Min-Max Coverage on graphs of bounded treewidth. This leaves a natural open problem — can the QPTAS be improved to a PTAS? The APX-hardness result for Asymmetric Min-Max Path Cover shows that such a PTAS would need to exploit the symmetries between different solution objects.

Acknowledgements

We thank Saket Saurabh for pointing us to the reference showing W[1]-hardness and ETH-hardness of Unary Bin Packing. Úrsula Hébert-Johnson and Daniel Lokshtanov were supported by NSF award CCF-2008838.

References

[1] D. Portugal, R. Rocha, MSP algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning, in: Proceedings of the 2010 ACM Symposium on Applied Computing, 2010, pp. 1271–1276.
[2] R. Borie, C. Tovey, S. Koenig, Algorithms and complexity results for graph-based pursuit evasion, Autonomous Robots 31 (4) (2011) 317–332.
[3] K. Easton, J. Burdick, A coverage algorithm for multi-robot boundary inspection, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, IEEE, 2005, pp. 727–734.
[4] E. Aaron, E. Kranakis, D. Krizanc, On the complexity of the multi-robot, multi-depot map visitation problem, in: 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, IEEE, 2011, pp. 795–800.
[5] B. Farbstein, A. Levin, Min–max cover of a graph with a small number of parts, Discrete Optimization 16 (2015) 51–61.
[6] E. M. Arkin, R. Hassin, A. Levin, Approximations for minimum and min-max vehicle routing problems, Journal of Algorithms 59 (1) (2006) 1–18.
[7] M. R. Khani, M. R. Salavatipour, Improved approximation algorithms for the min-max tree cover and bounded tree cover problems, Algorithmica 69 (2) (2014) 443–460.
[8] G. Even, N. Garg, J. Könemann, R. Ravi, A. Sinha, Min–max tree covers of graphs, Operations Research Letters 32 (4) (2004) 309–315.
[9] Z. Xu, Q. Wen, Approximation hardness of min–max tree covers, Operations Research Letters 38 (3) (2010) 169–173.

[10] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, Vol. 5, Springer, 2015.
[11] D. P. Williamson, D. B. Shmoys, The Design of Approximation Algorithms, Cambridge University Press, 2011.
[12] E. Aaron, D. Krizanc, E. Meyerson, Multi-robot foremost coverage of time-varying graphs, in: International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, Springer, 2014, pp. 22–38.
[13] M. Lampis, Parameterized approximation schemes using graph widths, in: International Colloquium on Automata, Languages, and Programming, Springer, 2014, pp. 775–786.
[14] H. L. Bodlaender, T. Hagerup, Parallel algorithms with optimal speedup for bounded treewidth, SIAM Journal on Computing 27 (6) (1998) 1725–1746.