On Covering Segments with Unit Intervals

Dan Bergren
School of Computing, DePaul University, Chicago, USA
bergren2@gmail.com

Eduard Eiben
Department of Computer Science, Royal Holloway, University of London, Egham, UK
eduard.eiben@rhul.ac.uk

Robert Ganian
Algorithms and Complexity Group, Vienna University of Technology, Vienna, Austria
rganian@gmail.com

Iyad Kanj
School of Computing, DePaul University, Chicago, USA
ikanj@cs.depaul.edu

Abstract

We study the problem of covering a set of segments on a line with the minimum number of unit-length intervals, where an interval covers a segment if at least one of the two endpoints of the segment falls in the unit interval. We also study several variants of this problem.

We show that the restrictions of the aforementioned problems to the set of instances in which all the segments have the same length are NP-hard. This result implies several NP-hardness results in the literature for variants and generalizations of the problems under consideration.

We then study the parameterized complexity of the aforementioned problems. We provide tight results for most of them by showing that they are fixed-parameter tractable for the restrictions in which all the segments have the same length, and are W[1]-complete otherwise.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Computational geometry

Keywords and phrases Segment covering, unit intervals, NP-completeness, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2020.13

Funding *Robert Ganian*: Robert Ganian acknowledges support by the Austrian Science Fund (FWF, project P31336).

1 Introduction

Problem Definition and Motivation. The problem of covering a set of points on the (real) line with the minimum number of closed unit-length intervals is a classical problem that can be solved in polynomial time by a simple greedy algorithm (e.g., see exercise 16.2-5 in [8] and exercise 5 in chapter 4 of [18]). A generalization of the above problem to that of covering a set of segments on the real line with the minimum number of unit intervals, where an interval *covers* a segment if at least one endpoint of the segment is in the interval, has been studied in several works [2, 3, 4]. For clarity, throughout the paper, we distinguish the entities to be covered from those used for covering, by referring to the former as *segments* and the latter as *intervals*. It is easy to see that the greedy algorithm – referred to above – no longer works for this generalization. In fact, this generalization turns out to be NP-hard, even though a straightforward (polynomial-time) greedy algorithm works for the restriction in which all segments have length at most 1 (unit).

Recently, several variants and generalizations of the above segment covering problem have been considered (as discussed below). Two natural variants arise based on whether the unit intervals can be arbitrarily chosen on the line, or are restricted to a given input set; the former version has been referred to as the *continuous* version as opposed to the latter *discrete* version. Moreover, a more restricted notion of covering has been studied as well, that we refer to henceforth as *exact covering*, in which exactly one endpoint from each segment must be covered by the unit intervals.

In this paper, we study the classical and parameterized complexity of the variants of segment covering by unit intervals discussed above, in most cases providing tight characterizations. The problem variants we study are: Continuous Segment Covering (Cont-SC); Discrete Segment Covering (Disc-SC); Continuous Exact Segment Covering (Cont-Exact-SC); and Discrete Exact Segment Covering (Disc-Exact-SC).

Related Work. Arkin et al. [2, 4] studied the exact covering problem of a set of *color classes*, where each color class contains two points on the real line of the same color, with the minimum number of intervals; here an interval covers a color class if it covers exactly one point from the color class. This is precisely the notion of exact segment covering, in which each color class corresponds to a segment whose endpoints are the two points in the class. It was shown in [2] that the aforementioned problem is NP-hard, and that the case in which the intervals are restricted to be unit intervals is NP-hard as well.

Arkin et al. [3, 4] also studied the problem of finding a *conflict-free* covering, where a color class can have both points covered, but it is not allowed to use an interval that covers both points of any color class. They showed that both the discrete and continuous versions of the aforementioned problem are NP-hard, and gave approximation algorithms of ratios 3 and 2, respectively, for them. They also studied a problem variant in which each color class consists of a horizontal or a vertical unit-length segment in the plane, and the goal is to compute a minimum-cardinality set of axes-parallel unit squares such that exactly one point from each segment is covered by the unit squares. They showed that this variant is NP-hard, and gave an approximation algorithm of ratio 6 for it. Achaaryya et al. [1] studied several variants of covering segments with axes-parallel unit squares in the plane. They obtained approximation algorithms and showed the NP-hardness of the variant in which all segments are horizontal unit segments.

The Cont-Exact-SC and Disc-Exact-SC problems under consideration are also related to an NP-hard combinatorial problem, referred to as the "Paintshop" problem [5, 15], that has applications in automotive industry. Other applications of covering line segments (referred to as "stabbing") with geometric objects (such as unit disks/squares) are in the area of networks security (see [1, 19]).

Finally, we mention that there is a vast amount of literature on other notions of covering and stabbing of geometric objects [7, 12, 13, 20, 21, 22].

Our Results. Our results for the Cont-SC, Disc-SC, Cont-Exact-SC, and Disc-Exact-SC problems can be summarized as follows (recall that the covering elements in all these segment covering variants are unit intervals):

(i) The restrictions of Cont-SC, Disc-SC, Cont-Exact-SC, and Disc-Exact-SC to instances in which all segments have the same length are NP-hard.
This NP-hardness result has important implications. First, it strengthens and implies several NP-hardness results in the literature about segment covering. The NP-hardness of Cont-Exact-SC implies the NP-hardness result stated in Theorem 6 of [2]. Moreover,

since we (can) assume that the uniform segment length in these restrictions of Cont-SC and Disc-SC is more than 1 (otherwise, the problem is polynomial-time solvable by a simple greedy algorithm), our NP-hardness result for Disc-SC implies the NP-hardness result in Theorem 1 of [3] (since the segment length is more than 1 unit and the intervals are unit intervals, the covering obtained is automatically a conflict-free covering). Second, the NP-hardness results for Cont-SC, Disc-SC, Cont-Exact-SC, and Disc-Exact-SC refine the complexity of these problems. For Cont-SC and Disc-SC, we already know that the slices of these problems consisting of instances in which each segment has length at most 1 unit are solvable in polynomial time by a greedy algorithm. (Note that we do not know if the same holds for Cont-Exact-SC and Disc-Exact-SC, as we do not know the complexity of their restrictions to instances in which each segment has length at most 1.) The above result shows that the slices of Cont-SC, Disc-SC, Cont-Exact-SC, and Disc-Exact-SC, consisting of instances in which all segments have the same length, are NP-hard.

The crucial insight required for our NP-hardness results is that, while the problems are one-dimensional, instances where the length of all segments differs significantly from the length of all intervals in fact behave like two-dimensional objects. We employ this in our proof by devising a series of two reductions, where we begin by considering a 2-dimensional segment covering problem whose instances are "nicely" embedded on a grid. We show that this aforementioned problem is NP-hard via a reduction from the restriction of Planar Vertex Cover to instances that are also "nicely" embedded on a grid. It is worth noting that, while the idea of proving NP-hardness by reducing from a problem with nice embedding properties (e.g., Planar 3-SAT) has been used in previous work [1, 4], the presented reduction stands out due to requiring complex "modularly constructed gadgets". We compose the above reduction with a second one that maps the segment covering problem on the grid to our 1-dimensional segment covering problems.

(ii) We show that the restrictions of Cont-SC, Disc-SC, Cont-Exact-SC, and Disc-Exact-SC to instances in which all segments have the same length are fixed-parameter tractable (FPT). The FPT algorithm for Cont-SC combines several algorithmic ideas. (The other FPT algorithms are similar.) It starts by computing an approximate solution of ratio 3 whose intervals contain *all* (input) segment endpoints. The algorithm then branches on all possibilities to determine how the approximate solution "interacts" with an optimal solution. Based on the determined interaction, the algorithm identifies endpoints of segments in the approximate solution, called *anchors*, around which the intervals in an optimal solution are *anchored* (i.e., placed). The goal then becomes to assign the endpoints of the segments to the anchors, where assigning an endpoint to an anchor means that the endpoint (and hence the associated segment) is covered by the interval in the optimal solution placed around that anchor. The algorithm then exploits the restriction that all segments have the same length, to define a domination relation among the anchors affecting each segment, which is then revealed through further branching. With these domination relations revealed, the resulting problem can be modeled as an instance of 2-Sat, which is solvable in polynomial time.

(iii) We show that Disc-SC and Cont-SC are W[1]-complete. Membership in W[1] is proved using the characterization of W[1] by Chen et al. [6], whereas the W[1]-hardness is proved via an FPT-reduction from the Multicolored Clique problem.

This reduction is quite involved, requiring gadget constructions that extend beyond the standard toolkit used in conventional W[1]-hardness reductions from Multicolored

Clique. The W[1]-completeness results, in conjunction with the results in (ii) above, provide tight results for the parameterized complexity of Disc-SC and Cont-SC. Note that, while the restrictions of Disc-SC and Cont-SC to instances in which all segments have equal length have the same classical complexity as their general counterparts, these restrictions exhibit a different behavior in terms of their parameterized complexity. The parameterized complexity of Cont-Exact-SC and Disc-Exact-SC remains open.

2 Preliminaries

We assume familiarity with the basic notation and terminology used in graph theory and parameterized complexity. We refer the reader to the standard books [10, 11] for more information on these subjects. The asymptotic notation $\mathcal{O}^*$ suppresses a polynomial factor in the input length. For $r \in \mathbb{N}$, we write $[r]$ as shorthand for the set $\{1, \ldots, r\}$.

We provide a brief overview of the basic parameterized complexity terminology used throughout the paper. A *parameterized problem* Q is a subset of $\Omega^* \times \mathbb{N}$, where Ω is a fixed alphabet. Each instance of Q is a pair (x, k), where $k \in \mathbb{N}$ is called the *parameter*. The problem Q is called *fixed-parameter tractable* (FPT) if it can be solved in time $f(k) \cdot |x|^{\mathcal{O}(1)}$, where f is a computable function. On the other hand, showing that a parameterized problem Q is hard for the parameterized complexity class W[1] provides strong conditional evidence that Q is not FPT. This is usually done by obtaining a suitable *FPT-reduction*, i.e., a reduction which runs in time $f(k) \cdot |x|^{\mathcal{O}(1)}$ and where the parameter of the output instance is upper-bounded by a function of the parameter of the input instance. We refer to the books by Downey and Fellows [11] and Cygan et al. [9] for an in-depth introduction to parameterized complexity.

2.1 Segment Covering Problems

The following problem will serve as a baseline for our problem definitions.

Discrete Segment Covering (Disc-SC)
Given: A set Γ of n intervals (called *segments* from here on out) on the rational line; a set $\mathcal{I}$ of unit-intervals on the rational line; $k \in \mathbb{N}$.
Parameter: k.
Question: Can the segments in Γ be covered by at most k intervals from $\mathcal{I}$?

Recall that an interval I *covers* a segment S if at least one endpoint of S lies in I. The Continuous Segment Covering problem (Cont-SC) is defined analogously to Disc-SC, with the sole distinction that the intervals can be chosen arbitrarily (i.e., there is no set $\mathcal{I}$ restricting which intervals may be chosen). For both of these problems, we also consider their *exact* versions, where we require that each segment also has an endpoint that is not contained in any interval (i.e., each segment must have precisely one "covered endpoint"); we call the associated problems Cont-Exact-SC and Disc-Exact-SC.

Finally, we denote by Disc-Equal-SC, Cont-Equal-SC, Disc-Equal-Exact-SC and Cont-Equal-Exact-SC the restrictions of Disc-SC, Cont-SC, Disc-Exact-SC and Cont-Exact-SC, respectively, to instances in which all segments in Γ have the same length. The restrictions of Cont-Equal-SC and Disc-Equal-SC to instances in which the length of the segments is at most 1 unit can be easily solved in polynomial time using a greedy approach; therefore, we will assume throughout this paper that the length of the segments in the instances of Disc-Equal-SC and Cont-Equal-SC is more than 1 unit.

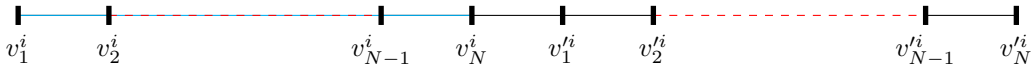

Figure 1 Illustration for the vertex-gadget construction. The segment $S_i = [v_1^i, v_N^i]$ is shown in cyan color.

▶ **Remark 1.** *There are polynomial-time FPT-reductions from* Cont-SC *and* Cont-Exact-SC *to* Disc-SC *and* Disc-Exact-SC*, respectively. The reduction from* Cont-SC *to* Disc-SC *follows from the fact that, for an instance of* Cont-SC*, we can always assume that the left endpoint of each covering unit-interval is an endpoint of a segment; if this is not the case for a covering unit-interval, we can shift it to the right until its left endpoint coincides with a segment's endpoint.*

3 Parameterized Complexity of Disc-SC and Cont-SC

In this section, we give a very high-level sketch of the W[1]-completeness proofs for Disc-SC and Cont-SC. Membership in W[1] is proved using the characterization of W[1] by Chen et al. [6]. The W[1]-hardness of Disc-SC is easier to explain as the set of covering unit-intervals is restricted; the proof can then be modified in order to lift this restriction, and obtain a W[1]-hardness proof for Cont-SC as well.

We show the W[1]-hardness of Disc-SC via an FPT-reduction from the W[1]-hard problem Multi-Colored Clique: Given a graph G with a proper k-coloring of its vertices, where each color class has cardinality N, decide if there exists a clique $Q \subseteq V(G)$ of size k [16, 9]; the parameter is k.

The reduction involves constructing three types of gadgets: vertex-selection gadgets, edge-verification gadgets, and edge-synchronization gadgets. Vertex-selection gadgets encode that k colorful vertices (i.e., no two vertices have the same color) in G are selected, and the edge-verification and edge-synchronization gadgets encode that the selected vertices form a clique. All the intervals and segments in the construction lie on the same (horizontal) line. The vertex and edge gadgets are placed far apart on the line, such that for any two gadgets, no two of their intervals overlap.

Vertex Gadgets. For each color class $C_i = \{v_1^i, \ldots, v_N^i\}, i \in [k]$, we place a sequence $\mathcal{S}^i$ of N interleaved unit intervals $[v_r^i, v_r'^i]$, $r \in [N]$, on the line, where the starting point of each interval is separated from the starting point of the next by a distance of $1/N$. We add the intervals in $\mathcal{S}^i$, for $i \in [k]$, to $\mathcal{I}$ as covering unit-intervals. For each sequence $\mathcal{S}^i$, we add the segment $S_i = [v_1^i, v_N^i]$ to Γ, which ensures that any solution must contain at least one interval from $\mathcal{S}^i$, in order to cover S_i. See Figure 1 for illustration.

Edge Gadgets. For each set of edges E_{ij}, between color classes C_i and C_j, where $i < j \in [k]$, let $m_{ij} = |E_{ij}|$. We place two interleaved sequences of unit-intervals. The construction of the two sequences is identical, and is done as follows. Set $M = m_{ij}$. The first sequence $\mathcal{S}_{ij}^1$ consists of unit-intervals $[e_r^1, e_r'^1]$, $r \in [M]$, such that $|e_r^1 e_{r+1}^1| = 1/M$, for $r \in [M-1]$; that is, the left endpoints of two consecutive intervals in this sequence are at distance $1/M$. Similarly, $\mathcal{S}_{ij}^2$ consists of unit-intervals $[e_r^2, e_r'^2]$, $r \in [M]$, such that $|e_r^2 e_{r+1}^2| = 1/M$, for $r \in [M-1]$. We place $\mathcal{S}_{ij}^1$ and $\mathcal{S}_{ij}^2$ on the line in an interleaving fashion, such that $[e_r^2, e_r'^2]$ in $\mathcal{S}_{ij}^2$ starts $1/(2M)$ units after $[e_r^1, e_r'^1]$ in $\mathcal{S}_{ij}^1$ ends, for $r \in [M]$. Put it differently, the sequence $\mathcal{S}_{ij}^2$ is shifted $1 + 1/(2M)$ units to the right from $\mathcal{S}_{ij}^1$. The reason behind this placement is that, if

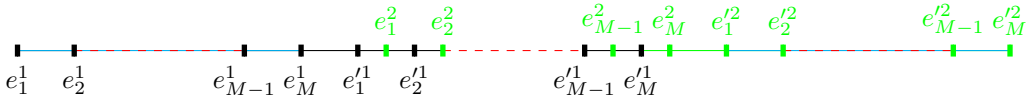

Figure 2 Illustration for the edge-gadget construction. The two segments $S_{ij}^1 = [e_1^1, e_M^1]$ and $S_{ij}^2 = [e_1'^2, e_M'^2]$ are shown in cyan color.

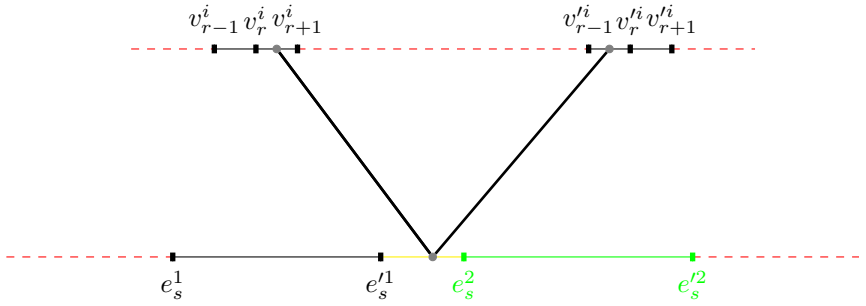

Figure 3 Illustration for the connection between a vertex-gadget and an edge-gadget that encodes vertex-edge incidency. Only the relevant portions of the gadgets are shown, and for clarity, the two gadgets are drawn on top of one another, rather than on the same line.

we assume that copies of the same interval are chosen from $\mathcal{S}_{ij}^1, \mathcal{S}_{ij}^2$, a property that will be ensured by the edge-synchronization gadgets discussed below, then the $1/(2M)$ units gap between an interval $[e_r^1, e_r'^1]$ in $\mathcal{S}_{ij}^1$ and its copy $[e_r^2, e_r'^2]$, $r \in [M]$, in $\mathcal{S}_{ij}^2$ is covered by *any* choice of an interval from $\mathcal{S}_{ij}^1$ and its copy in $\mathcal{S}_{ij}^2$, *except* the choice of $[e_r^1, e_r'^1]$ and $[e_r^2, e_r'^2]$. This property is crucial for encoding vertex-edge incidences. We add the unit-intervals of $\mathcal{S}_{ij}^1$ and $\mathcal{S}_{ij}^2$ to $\mathcal{I}$ as covering unit-intervals. We add the two segments $S_{ij}^1 = [e_1^1, e_M^1]$ and $S_{ij}^2 = [e_1'^2, e_M'^2]$ to Γ; these two segments ensure that any solution must contain at least one interval from each of $\mathcal{S}_{ij}^1$ and $\mathcal{S}_{ij}^2$. See Figure 2 for illustration.

Edge-synchronization Gadgets. For each edge-gadget consisting of two sequences $\mathcal{S}_{ij}^1$ and $\mathcal{S}_{ij}^2$ of unit-intervals, we construct a sequence of interleaved unit-intervals, $\mathcal{S}_{ij}^3$, that is constructed identically to $\mathcal{S}_{ij}^1$ and $\mathcal{S}_{ij}^2$. We add the intervals in $\mathcal{S}_{ij}^3$ to $\mathcal{I}$ as covering unit-intervals. We add the segment $S_{ij}^3 = [e_1^3, e_M^3]$ to Γ, which ensures that any solution must contain at least one interval from $\mathcal{S}_{ij}^3$. The intervals in $\mathcal{S}_{ij}^3$ will ensure that, in the desired solution, three copies of the same interval are chosen, one from each of $\mathcal{S}_{ij}^1$, $\mathcal{S}_{ij}^2$, and $\mathcal{S}_{ij}^3$.

Connecting the Gadgets. We encode vertex-edge incidences in G by adding segments between vertex-gadgets and corresponding edges-gadgets. For a vertex v_r, $r \in [N]$, in color class C_i (resp. C_j) in G, and an edge e_{ij} incident to v_r and to some vertex in color class C_j (resp. C_i), let $[e_s^1, e_s'^1]$ and $[e_s^2, e_s'^2]$ be the intervals corresponding to e_{ij} in $\mathcal{S}_{ij}^1$ and $\mathcal{S}_{ij}^2$, respectively. Do the following (see Figure 3 for an illustration): (if $r < N$) create a segment with one endpoint in the interval (v_r^i, v_{r+1}^i) in $\mathcal{S}^i$ (resp. in (v_r^j, v_{r+1}^j)), and the other in $(e_s'^1, e_s^2)$; and (if $r > 1$) create a segment with one endpoint in $(v_{r-1}'^i, v_r'^i)$ in $\mathcal{S}^i$ (resp. in $(v_{r-1}'^j, v_r'^j)$ in $\mathcal{S}^j$), and the other in $(e_s'^1, e_s^2)$.

We encode edge-synchronization for each triplet of sequences $\mathcal{S}_{ij}^1$, $\mathcal{S}_{ij}^2$, and $\mathcal{S}_{ij}^3$, corresponding to the edges in E_{ij}, where $|E_{ij}| = m_{ij}$, as follows (see Figure 4 for an illustration). For each $s \in [m_{ij} - 1]$: (i) create a segment with one endpoint in the interval (e_s^1, e_{s+1}^1) and the other in $(e_s'^3, e_{s+1}'^3)$; (ii) create a segment with one endpoint in (e_s^2, e_{s+1}^2) and the other in

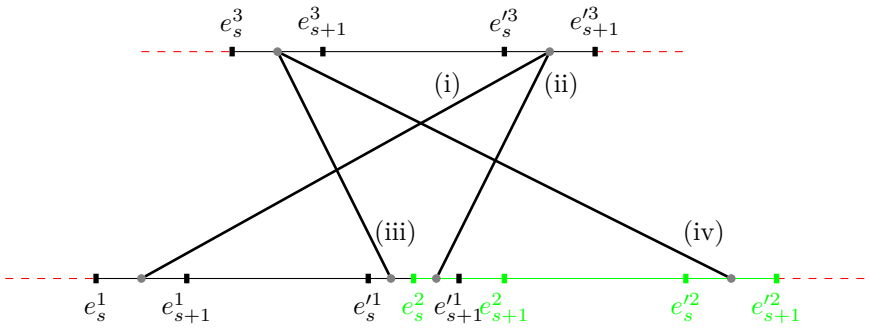

Figure 4 Illustration of the edge-synchronization gadget construction. The four types of the inter-sequence segments are indicated. For clarity, $\mathcal{S}^3_{ij}$ is drawn on top of $\mathcal{S}^1_{ij}$ and $\mathcal{S}^2_{ij}$.

(e'^3_s, e'^3_{s+1}); (iii) create a segment with one endpoint in (e^3_s, e^3_{s+1}) and the other in (e'^1_s, e'^1_{s+1}); and (iv) create a segment with one endpoint in (e^3_s, e^3_{s+1}) and the other in (e'^2_s, e'^2_{s+1}).

We can now show that (G, k) is a YES-instance of Multi-Colored Clique if and only if $(\Gamma, \mathcal{I}, k')$, where $k' = k + 3\binom{k}{2}$, is a YES-instance of Disc-SC. We conclude with:

▶ **Theorem 2.** *Disc-SC is* W[1]*-complete.*

▶ **Corollary 3.** *Cont-SC is* W[1]*-complete.*

4 NP-Completeness of the Equal Segment-Length Variants

In this section we show that all of our considered problems are NP-complete even when restricted to the case where all segments have equal length (i.e., Disc-Equal-SC, Cont-Equal-SC, Disc-Equal-Exact-SC and Cont-Equal-Exact-SC). We do so via a two-step reduction through the following intermediate problem:

Grid Segment Covering
Given: A set Γ of n vertical segments, each of length 1, with endpoints on a $q \times q$ grid with $q \leq 100 \cdot n$; $k \in \mathbb{N}$.
Parameter: k.
Question: Can the segments in Γ be covered by at most k horizontal segments of length 2?

For consistency with the terminology used in this paper (in the definition of segment covering problems under consideration), we will abuse the notation and refer to the horizontal (covering) segments of length 2 as intervals, and to the vertical segments of length 1 (to be covered) as segments. We define the Exact Grid Segment Covering analogously to Grid Segment Covering, with the sole distinction being that each segment in Γ must have precisely one endpoint covered by the solution (i.e., the set of intervals). The main technical obstacle on the way to the NP-hardness of our problems lies in showing that Grid Segment Covering and Exact Grid Segment Covering are NP-hard. The following theorem will serve as a starting point towards obtaining these results.

▶ **Theorem 4** (Theorem 5.9 of [17])**.** *Given a planar graph G of degree at most 3 with $n > 4$ vertices, there is a linear time algorithm that constructs a plane orthogonal drawing of G on an $\lfloor \frac{n}{2} \rfloor \times \lfloor \frac{n}{2} \rfloor$ grid with at most $\lfloor \frac{n}{2} \rfloor + 1$ bends[1], and with the property that there is a spanning tree of $n - 1$ straight-line edges, while all nontree edges have at most one bend.*

[1] A *bend* is the meeting point of a horizontal and a vertical line in the drawing of an edge.

4.1 Grid Segment Covering is NP-Complete

We reduce from a restriction of the NP-complete problem [14] Planar 3-Vertex Cover (i.e., Vertex Cover restricted to planar graphs of maximum degree 3). We first show that this restriction remains NP-complete:

▶ **Theorem 5.** *Planar 3-Vertex Cover is NP-hard even on instances with n vertices and a plane orthogonal drawing on an $n \times n$ grid, with no bends, even when such an embedding is provided as input.*

With Theorem 5 in hand, we can proceed to the description of the reduction strategy. Given an n-vertex instance (G, ℓ) of Planar 3-Vertex Cover, the first step is to invoke Theorem 5 to obtain a plane orthogonal drawing Ω of G on an $n \times n$ grid satisfying the property that every edge is a horizontal or a vertical line segment in this grid. Next, we refine the grid underlying Ω by a factor of 100 – more formally, we replace each cell in the grid underlying Ω with a 100×100 subgrid.

We first outline the reduction. The reduction will represent each vertex v in G with a *vertex gadget* $\alpha(v)$. This gadget consists of a set of segments placed along the border of a geometric object that is roughly centered around the position of v in Ω, and that extend in the directions of the edges incident to v. These vertex gadgets will have the following properties:

- **Vertical connections**: If v and w are adjacent vertices in G, then there is an "interface" spanning a 2×3 subgrid such that either $\alpha(v)$ is placed at the bottom right of the subgrid and $\alpha(w)$ at the top left, or vice versa. This property will be used by the *edge gadget* $\beta(uv)$.
- **Duality of choice**: There are two "optimal configurations" of intervals that allow us to cover all segments in $\alpha(v)$: one uses the minimum number of intervals required but does not help us cover the segments located in the edge gadgets connected to $\alpha(v)$, while the other requires one extra interval but also helps us cover segments in the edge gadgets connected to $\alpha(v)$. These optimal configurations cover each segment in $\alpha(v)$ only once.

The following lemma formalizes the precise properties we require from the vertex gadgets. Further intuition about the construction of the gadgets is provided in Figure 5.

▶ **Lemma 6.** *Given G, ℓ, Ω as above, in polynomial time we can construct a mapping α from the vertex set $V(G)$ that maps each $v \in V(G)$ to a gadget $\alpha(v)$. Each such gadget $\alpha(v)$ consists of a set of $|\alpha(v)|$ segments and up to 3 "link" points, each corresponding to an edge incident to v, with the following properties:*

1. *Any interval that can cover segments from $\alpha(v)$ cannot cover segments from any $\alpha(w)$ for $w \neq v$.*
2. *There exists no set of intervals of size less than $\mathsf{cost}(\alpha(v)) = \frac{|\alpha(v)|-1}{2}$ that covers all segments in $\alpha(v)$; moreover, there exists a set of intervals of size $\mathsf{cost}(\alpha(v))$ which is an exact covering of all segments in $\alpha(v)$.*
3. *There exists no set of intervals of size less than $\mathsf{cost}(\alpha(v)) + 1$ that covers all segments in $\alpha(v)$ together with at least one link point of $\alpha(v)$; moreover, there exists a set of intervals of size $\mathsf{cost}(\alpha(v)) + 1$ which is an exact covering of all segments in $\alpha(v)$ and additionally covers all link points of $\alpha(v)$.*
4. *For each edge $vw \in E(G)$ with two link points (x_v, y_v) and (x_w, y_w), it holds that either $x_v = x_w + 2$ and $y_v = y_w - 3$ or vice-versa.*

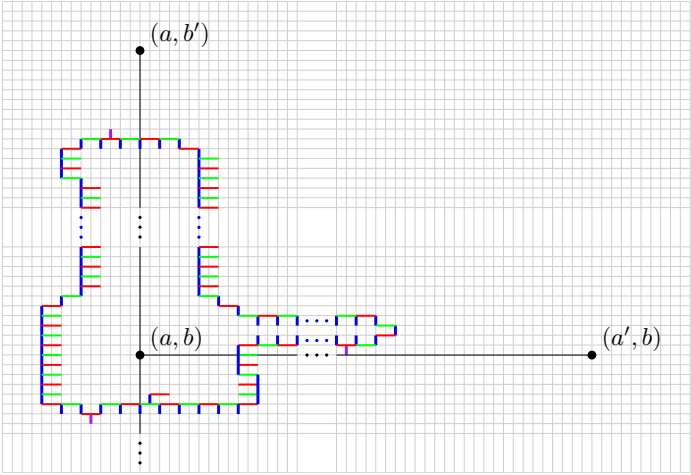

Figure 5 A vertex gadget (blue) for a vertex v located at point (a, b). Link points are marked by purple segments. The set of green intervals has size $\mathsf{cost}(\alpha(v))$ and exactly covers $\alpha(v)$. The set of red interval has size $\mathsf{cost}(\alpha(v)) + 1$ and covers $\alpha(v)$ and all the link points.

After applying Lemma 6 to construct the vertex gadgets, we will construct, for each edge vw, an edge gadget $\beta(vw)$. This will consist of the segments $[(x_v, y_v), (x_v, y_v - 1)]$, $[(x_v + 1, y_v - 1), (x_v + 1, y_v - 2)]$, and $[(x_v + 2, y_v - 2), (x_v + 2, y_v - 3)]$.

We now proceed to the NP-hardness proof.

▶ **Theorem 7.** *Grid Segment Covering and Exact Grid Segment Covering are NP-complete.*

Proof Sketch. Inclusion in NP is trivial. To show NP-hardness, we reduce from Planar 3-Vertex Cover. Given an instance (G, ℓ) of Planar 3-Vertex Cover, we apply the construction: notably, we use Theorem 4 to obtain an embedding of G on an orthogonal grid, refine this grid by a factor of 100, and replace all vertices with vertex gadgets as per Lemma 6, and all edges with edge-gadgets. Set $k = \ell + |E(G)| + \sum_{v \in V(G)} \mathsf{cost}(\alpha(v))$. Now it suffices to show that (G, ℓ) is a YES-instance if and only if (Σ, k) is a YES-instance. ◀

4.2 Reductions from the Grid Segment Covering Problem

▶ **Theorem 8.** *Disc-Equal-SC, Cont-Equal-SC, Disc-Equal-Exact-SC and Cont-Equal-Exact-SC are NP-complete.*

Proof Sketch. We sketch the reduction for Cont-Equal-SC. Let (Γ, k) be an instance of Grid Segment Covering with endpoints on $q \times q$ grid. We construct an instance (Γ', k') of Cont-Equal-SC as follows. For a segment $I \in \Gamma$ with endpoints (w, h) and $(w, h + 1)$, we add in Γ' the segment $[\frac{(q+3)h+w}{2}, \frac{(q+3)(h+1)+w}{2}]$ and we let $k' = k$. Note that if a segment $I' \in \Gamma'$ has one endpoint at $b \in \mathbb{Q}$, then there exist $h \in \mathbb{N}$ and $w \in \mathbb{Q}$ (with $0 \leq w \leq q$) such that $b = \frac{(q+3)h+w}{2}$. Hence, if a unit interval covers endpoints $b_1 = \frac{(q+3)h_1+w_1}{2}$ and $b_2 = \frac{(q+3)h_2+w_2}{2}$, then it follows that $h_1 = h_2$, because otherwise $|b_2 - b_1| \geq \frac{3}{2}$.

To conclude the proof for Cont-Equal-SC, it suffices to show that (Γ', k') is YES-instance of Cont-Equal-SC if and only if (Γ, k) is YES-instance of Grid Segment Covering. The construction is identical for Disc-Equal-Exact-SC, with the sole distinction being the use of Exact Grid Segment Covering. The other two results follow from Remark 1. ◀

5 FPT Algorithms for the Equal Segment-Length Variants

In this section, we give FPT algorithms for Cont-Equal-SC, Cont-Equal-Exact-SC, Disc-Equal-SC, and Disc-Equal-Exact-SC. Before proceeding to the technical details, we first discuss and give an overview of the FPT algorithm for Cont-Equal-SC.

Let (Γ, k) be an instance of Cont-Equal-SC. The FPT algorithm starts by computing an approximate solution, $\mathcal{S}_{apx}$, for Γ of size at most $3k$ (assuming that a solution of size k exists) whose intervals contain *all* endpoints of the segments in Γ. The algorithm then guesses (i.e., branches on all possibilities) how $\mathcal{S}_{apx}$ interacts with a solution, $\mathcal{S}_{opt}$, for Γ of size at most k. Based on this guess, the algorithm identifies endpoints of segments in $\mathcal{S}_{apx}$, called *anchors*, around which the intervals in $\mathcal{S}_{opt}$ are *anchored* (i.e., placed). The goal then becomes to assign the endpoints of the segments in Γ to the guessed anchors, where assigning an endpoint to an anchor means that the endpoint (and hence the associated segment) is covered by the interval in $\mathcal{S}_{opt}$ placed around that anchor, and then modeling the problem as an instance of 2-Sat that stipulates that, for each segment, at least one of its endpoints is covered by a unit-interval placed around an anchor. The issue, however, is that for a segment, there could be four anchors whose intervals cover its (two) endpoints, and this cannot be stipulated by size-2 clauses. This issue is resolved by exploiting the crucial property that all segments have the same length, which enables us to define a notion of domination among the anchors that could potentially cover the same segment, and guess this domination. Once the domination relations among the anchors affecting each segment are revealed, encoding the covering requirement using size-2 clauses becomes possible, as the number of anchors affecting each segment can be reduced from 4 to 2. Extra clauses are then added to the 2-Sat instance to enforce that the assignment corresponds to a proper placement of k unit-length covering intervals that cover the segments in Γ. We proceed to the details.

Let (Γ, k) be an instance of Cont-Equal-SC. We start with the following simple result:

▶ **Fact 9.** *In $\mathcal{O}(|\Gamma| \log |\Gamma|)$ time, we can compute a solution $\mathcal{S}_{apx}$ to Γ that is within ratio 3 from an optimal solution and that contains all endpoints of the segments in Γ.*

Proof. For a unit-length interval I, define the *left dual* (resp. *right dual*) of I, denoted, $\overline{I_L}$ (resp. $\overline{I_R}$), to be the interval that is the translation of I (along the horizontal line) to the left (resp. to the right) by a vector whose length is equal to the length of the segments in Γ. Observe that the set of segments in Γ whose right (resp. left) endpoints are covered by a unit-length interval I is the same set of segments whose left (resp. right) endpoints are covered by $\overline{I_L}$ (resp. $\overline{I_R}$).

The approximation algorithm, denoted APX-ALGO, finds a set of unit-length intervals of minimum cardinality that covers the endpoints of all the segments in Γ; the problem of covering a set of N points on a line by the minimum number of unit-length intervals is known to be solvable in $\mathcal{O}(N \lg N)$ time by a greedy approach (e.g., see problem 16.2-5 in [8]).

Consider now an optimal solution for Γ, and for each interval I in the optimal solution, add both its left and right duals $\overline{I_L}$ and $\overline{I_R}$. We obtain a solution that contains all segment endpoints and whose cardinality is at most thrice that of the optimal solution. Since APX-ALGO produces an optimal solution for covering the endpoints of all segments in Γ, the result follows. ◀

Based on the above, if $|\mathcal{S}_{apx}| > 3k$, the instance (Γ, k) is a no-instance of Cont-Equal-SC. Assume henceforth that $|\mathcal{S}_{apx}| \leq 3k$. Every endpoint of a segment in Γ is contained in an interval of $\mathcal{S}_{apx}$. Without loss of generality, we can assume that the intervals in $\mathcal{S}_{apx}$ are pairwise disjoint, and that each starts at an endpoint of a segment in Γ (see also Remark 1).

If not, we can process the intervals in $\mathcal{S}_{apx}$ to ensure this property, while maintaining the property that every endpoint of a segment in Γ is contained in an interval of $\mathcal{S}_{apx}$. To do so, we repeatedly pick the leftmost two overlapping intervals in $\mathcal{S}_{apx}$, say I_r, I_s where I_r starts before I_s. We shift I_s to the right until it no longer overlaps with I_r, while starting at an endpoint in Γ and retaining the set of segment endpoints contained in $I_r \cup I_s$. If at some point this shifting results in an interval that is devoid of segment endpoints, we remove this interval from $\mathcal{S}_{apx}$. Clearly, at the end of this process, the set $\mathcal{S}_{apx}$ consists of pairwise-disjoint intervals that contain all endpoints of the segments in Γ, each of whose intervals starts at an endpoint of a segment in Γ, and satisfying $|\mathcal{S}_{apx}| \leq 3k$.

Let $\mathcal{S}_{opt}$ be an optimal solution for Γ, and assume that $|\mathcal{S}_{opt}| \leq k$. W.l.o.g., we will assume that $\mathcal{S}_{opt}$ is chosen so as to maximize the number of intervals that are common to both $\mathcal{S}_{opt}$ and $\mathcal{S}_{apx}$ (i.e., maximize $|\mathcal{S}_{opt} \cap \mathcal{S}_{apx}|$).

▶ **Definition 10.** An endpoint $\mathfrak{a}$ of an interval in $\mathcal{S}_{apx}$ is called an *anchor* if there is an interval I in $\mathcal{S}_{opt}$ that contains $\mathfrak{a}$, in which case we say that I *induces* $\mathfrak{a}$.

The FPT-algorithm for Cont-Equal-SC performs the following steps:

Step (1). Guessing the Anchors: The FPT-algorithm starts by guessing how $\mathcal{S}_{opt}$ interacts with $\mathcal{S}_{apx}$. First, it guesses the number k' of intervals that are common to both $\mathcal{S}_{apx}$ and $\mathcal{S}_{opt}$ (i.e., $|\mathcal{S}_{apx} \cap \mathcal{S}_{opt}|$). Then, the algorithm guesses the k' common intervals, removes them from $\mathcal{S}_{apx}$, and updates Γ and the parameter k accordingly (by removing from Γ all segments covered by the k' intervals, and setting $k = k - k'$). By the same arguments made above about $\mathcal{S}_{apx}$, we can assume from now on that the intervals in $\mathcal{S}_{opt}$ are disjoint, and that each starts at a segment endpoint. Every interval $I \in \mathcal{S}_{opt}$ intersects two consecutive intervals in $\mathcal{S}_{apx}$. Otherwise, if I intersects only one interval $I' \in \mathcal{S}_{apx}$, since each interval in $\mathcal{S}_{opt}$ starts at a segment endpoint, I would intersect I' only at the left endpoint of I, and all segment endpoints in I must also be in I'. This contradicts our choice of $\mathcal{S}_{opt}$ as an optimal solution that maximizes $|\mathcal{S}_{apx} \cap \mathcal{S}_{opt}|$ (since I could be shifted left to obtain an optimal solution containing I'). Hence, if I contains the left (resp. right) endpoint of I', then it must contain the right (resp. left) endpoint of the predecessor (resp. successor) interval of I' in $\mathcal{S}_{apx}$. Next, for each endpoint of an interval in $\mathcal{S}_{apx}$, the algorithm guesses whether it is an anchor. Let Υ be the set of guessed anchors.

Step (2). Restructuring the Anchors: From Step (1), if an anchor $\mathfrak{a} \in \Upsilon$ is the right (resp. left) endpoint of an interval $I' \in \mathcal{S}_{apx}$, then the interval $I \in \mathcal{S}_{opt}$ that induces $\mathfrak{a}$ intersects the successor (resp. predecessor) of I' in $\mathcal{S}_{apx}$, and hence, the left (resp. right) endpoint of the successor (resp. predecessor) of I' must be an anchor induced by I as well. If after Step (1) Υ does not conform to the above, then we can reject the guess, as there will be another guess that satisfies the above property. Based on this property, we will remove from Υ the anchors that are right endpoints of intervals in $\mathcal{S}_{apx}$. After removing these anchors, each interval in $\mathcal{S}_{opt}$ induces *exactly* one anchor in Υ that is the left endpoint of an interval in $\mathcal{S}_{apx}$. Since the intervals in $\mathcal{S}_{apx}$ are pairwise disjoint, any two anchors in Υ are more than 1-unit apart. Consequently, if $|\Upsilon| > k$, then we can reject the guess in Step (1), as no solution of size at most k realizing the guess exists.

Step (3). Domination among Anchors: Let $\mathfrak{a}, \mathfrak{b}$ be two anchors, and let $\mathcal{S} \subseteq \Gamma$. We say that $\mathfrak{a}$ *dominates* $\mathfrak{b}$ w.r.t. $\mathcal{S}$, written as $\mathfrak{a} \succeq_{\mathcal{S}} \mathfrak{b}$ or $\mathfrak{b} \preceq_{\mathcal{S}} \mathfrak{a}$, if the set of segments in $\mathcal{S}$ covered by (the interval in $\mathcal{S}_{opt}$ inducing) $\mathfrak{a}$ is a superset of the set of segments in $\mathcal{S}$ covered by (the

interval in $\mathcal{S}_{opt}$ inducing) $\mathfrak{b}$. For convenience, we will define the notion of an *empty anchor*, denoted $\otimes$; the set of segments covered by the empty anchor is the empty set ϕ, and hence, every anchor dominates $\otimes$. We will use the notion of domination, in conjunction with a guessing process, to reduce the instance (Γ, k), resulting from Steps (1) and (2) above, to an instance of 2-SAT, which can then be solved in polynomial time. To do so, we consider the intervals in $\mathcal{S}_{apx}$ from left to right, and construct an instance $\mathcal{F}$ of 2-SAT. We initialize $\mathcal{F}$ to the empty set, and we will add clauses to $\mathcal{F}$ as follows.

Let $I \in \mathcal{S}_{apx}$ be the interval currently under consideration (when scanning the intervals in $\mathcal{S}_{apx}$ from left to right), and let $\mathcal{S}$ be the set of segments whose left endpoints are on I. (We are not concerned at this point about the set of segments whose right endpoint are on I since those have been considered earlier in the process.) Observe that, since all segments in Γ have the same length, and all covering intervals have the same length, the right endpoints of the segments in $\mathcal{S}$ fall either on one or on two intervals in $\mathcal{S}_{apx}$. (Otherwise, there would be two segments whose left endpoints lie on I, and hence, of distance at most 1 unit, but whose right endpoints are more than one unit apart.) This property, which again stems from the fact that all segments in Γ have the same length, is *crucial*, and is the key idea behind the FPT algorithm, as will be seen below.

We treat the more complex case in which the right endpoints of the segments in $\mathcal{S}$ fall on two intervals I_1, I_2 in $\mathcal{S}_{apx}$, where I_2 is the successor of I_1 in $\mathcal{S}_{apx}$; the case where they fall on one interval is simpler, and is a subcase of the treated case, as we explain below. Partition $\mathcal{S}$ into $\mathcal{S}_1, \mathcal{S}_2$, where $\mathcal{S}_1$ consists of those segments in $\mathcal{S}$ whose right endpoints are on I_1, and $\mathcal{S}_2$ of those whose right endpoints are on I_2. Let $\mathfrak{a}$ be the left endpoint of I if it is an anchor, and $\mathfrak{a} = \otimes$ otherwise; let $\mathfrak{b}$ be the left endpoint of the successor of I in $\mathcal{S}_{apx}$ if its an anchor, and $\mathfrak{b} = \otimes$ otherwise; let $\mathfrak{u}$ be the left endpoint of I_1 if it is an anchor, and $\mathfrak{u} = \otimes$ otherwise; let $\mathfrak{s}$ be the left endpoint of I_2 if it is an anchor, and $\mathfrak{s} = \otimes$ otherwise; and let $\mathfrak{t}$ be the left endpoint of the successor of I_2 in $\mathcal{S}_{apx}$ if it is an anchor, and $\mathfrak{t} = \otimes$ otherwise. Observe that, since each of the anchors $\mathfrak{a}$ and $\mathfrak{u}$ covers a prefix (possibly empty) of the sequence of segments in $\mathcal{S}_1$ when ordered from left to right, one of the two anchors must dominate the other w.r.t. $\mathcal{S}_1$. Similarly, each of $\mathfrak{b}$ and $\mathfrak{s}$ covers a suffix (possibly empty) of the sequence of segments in $\mathcal{S}_1$, and hence one must dominate the other w.r.t. $\mathcal{S}_1$. With respect to $\mathcal{S}_2$, one of $\mathfrak{a}$ and $\mathfrak{s}$ must dominate the other, and one of $\mathfrak{b}$ and $\mathfrak{t}$ must dominate the other. Therefore, (i) either $\mathfrak{a} \preceq_{\mathcal{S}_1} \mathfrak{u}$ or $\mathfrak{u} \preceq_{\mathcal{S}_1} \mathfrak{a}$ and (ii) either $\mathfrak{b} \preceq_{\mathcal{S}_1} \mathfrak{s}$ or $\mathfrak{s} \preceq_{\mathcal{S}_1} \mathfrak{b}$; and w.r.t. the segments in $\mathcal{S}_2$, we have (iii) either $\mathfrak{a} \preceq_{\mathcal{S}_2} \mathfrak{s}$ or $\mathfrak{s} \preceq_{\mathcal{S}_2} \mathfrak{a}$, and (iv) either $\mathfrak{b} \preceq_{\mathcal{S}_2} \mathfrak{t}$ or $\mathfrak{t} \preceq_{\mathcal{S}_2} \mathfrak{b}$.

The algorithm now guesses, for each of Cases (i) – (iv) above, which anchor dominates the other. This guessing results in four cases w.r.t. each of $\mathcal{S}_1$ and $\mathcal{S}_2$ that are described below, and hence, results in sixteen cases overall. If the right endpoints of the segments in $\mathcal{S}$ fall on one interval I_1, then the guessing results only in the four cases w.r.t. $\mathcal{S}_1$ distinguished below (and anchor $\mathfrak{t}$ would not be needed). We will create Boolean variables corresponding to endpoints of segments in Γ. A Boolean variable of the form $x_{\mathfrak{h}}$, where x is an endpoint of a segment $S \in \Gamma$ and $\mathfrak{h}$ is an anchor, is true if and only if point x (and hence S) is covered by the interval inducing $\mathfrak{h}$. We will then form an instance of 2-SAT that encodes the instance (Γ, k) of CONT-EQUAL-SC under the assumed guess. For simplicity of the presentation, we make the following assumptions. If a Boolean variable $x_{\mathfrak{h}}$, associated with anchor $\mathfrak{h}$, is such that either $\mathfrak{h} = \otimes$, or the distance between x and $\mathfrak{h}$ is more than 1 unit (i.e., more than the length of a unit-length covering interval), then we set/fix the value of $x_{\mathfrak{h}}$ to FALSE. We start by associating with every endpoint x of a segment in Γ two Boolean variables as follows. Let $\mathfrak{h}$ and $\mathfrak{h}'$ be the two anchors directly to the left and right, respectively, of x. We associate

Figure 6 Illustration for **Cases 1-4** w.r.t. $\mathcal{S}_1$ and $\mathcal{S}_2$. The two red circles designate the endpoints x, y of a segment $S \in \mathcal{S}_1$, and the green circles designate the endpoints x', y' of a segment $S' \in \mathcal{S}_2$, where x is to the left of y and x' is to the left of y'. For clarity, only the endpoints of S, S' are shown. If the guess w.r.t. $\mathcal{S}_1$ is that $\mathfrak{a} \succeq \mathfrak{u}$ and $\mathfrak{s} \succeq \mathfrak{b}$ and w.r.t. $\mathcal{S}_2$ is that $\mathfrak{s} \succeq \mathfrak{a}$ and $\mathfrak{b} \succeq \mathfrak{t}$ then clauses $\{x_\mathfrak{a} \vee y_\mathfrak{s}\}$ and $\{x'_\mathfrak{b} \vee y'_\mathfrak{s}\}$ are added to $\mathcal{F}$.

the two Boolean variables $x_\mathfrak{h}$ and $x_{\mathfrak{h}'}$ with x. (Note that $\mathfrak{h}$ or $\mathfrak{h}'$ could be $\otimes$, or of distance more than 1 unit from x, and in which case the corresponding Boolean variable would be set to FALSE.) The four cases distinguished w.r.t. $\mathcal{S}_1$ are (see Figure 6 for illustration):

Case 1: $\mathfrak{a} \succeq \mathfrak{u}$ and $\mathfrak{b} \succeq \mathfrak{s}$. For each endpoint x on I such that x is the left endpoint of a segment in $\mathcal{S}_1$, add the clause $\{x_\mathfrak{a} \vee x_\mathfrak{b}\}$ to $\mathcal{F}$.

Case 2: $\mathfrak{a} \succeq \mathfrak{u}$ and $\mathfrak{s} \succeq \mathfrak{b}$. For each endpoint x on I such that x is the left endpoint of a segment S in $\mathcal{S}_1$, let y be the right endpoint of S. Add the clause $\{x_\mathfrak{a} \vee y_\mathfrak{s}\}$ to $\mathcal{F}$.

Case 3: $\mathfrak{u} \succeq \mathfrak{a}$ and $\mathfrak{b} \succeq \mathfrak{s}$. For each endpoint x on I such that x is the left endpoint of a segment S in $\mathcal{S}_1$, let y be the right endpoint of S. Add the clause $\{x_\mathfrak{b} \vee y_\mathfrak{u}\}$ to $\mathcal{F}$.

Case 4: $\mathfrak{u} \succeq \mathfrak{a}$ and $\mathfrak{s} \succeq \mathfrak{b}$. For each endpoint x on I such that x is the left endpoint of a segment S in $\mathcal{S}_1$, let y be the right endpoint of S. Add the clause $\{y_\mathfrak{u} \vee y_\mathfrak{s}\}$ to $\mathcal{F}$.

Similarly, we can distinguish four cases w.r.t. $\mathcal{S}_2$, based on the two domination relations in (iii) and (iv) discussed earlier, and add clauses to $\mathcal{F}$ accordingly.

After processing the intervals in $\mathcal{S}_{apx}$, guessing domination, associating Boolean variables, and adding clauses to $\mathcal{F}$, we add to $\mathcal{F}$ the following "enforcement" clauses. For every anchor $\mathfrak{a}$, and every two variables $x_\mathfrak{a}, z_\mathfrak{a}$, corresponding to segment endpoints x, z, respectively, associated with $\mathfrak{a}$:

(E1) If x and z are on the right (resp. left) side of $\mathfrak{a}$, and if z (resp. x) is to the right (resp. left) of x, add $\{\overline{z}_\mathfrak{a} \vee x_\mathfrak{a}\}$ (resp. $\{\overline{x}_\mathfrak{a} \vee z_\mathfrak{a}\}$) to $\mathcal{F}$; and (E2) if x and z are on opposite sides of $\mathfrak{a}$ and the distance between them is more than 1 unit, add $\{\overline{x}_\mathfrak{a} \vee \overline{z}_\mathfrak{a}\}$ to $\mathcal{F}$. The enforcement clauses ensure that any satisfying assignment to $\mathcal{F}$ corresponds to an assignment of segment endpoints to anchors satisfying: (1) All endpoints assigned to the same anchor can be covered by a unit interval (E2); and (2) if an endpoint of a segment S that is assigned to an anchor $\mathfrak{h}$ is covered by the interval $I_\mathfrak{h}$ inducing $\mathfrak{h}$, then any segment endpoint assigned to $\mathfrak{h}$ that is between that endpoint of S and $\mathfrak{h}$ is also covered by $I_\mathfrak{h}$ (E1).

The FPT algorithm accepts if any of the guesses it makes leads to a formula $\mathcal{F}$ that is a YES-instance of 2-SAT, and rejects otherwise. We obtain the following result:

▶ **Theorem 11.** *CONT-EQUAL-SC can be solved in time* $\mathcal{O}((2^4 \cdot 3 \cdot e^{5/3})^k \cdot n \log n) = \mathcal{O}(2^{8k} \cdot n \log n)$*, where* $n = |\Gamma|$*, and hence is FPT.*

Proof. We first argue the correctness of the algorithm. The instance (Γ, k) is a YES-instance of CONT-EQUAL-SC if and only if there exists an optimal solution $\mathcal{S}_{opt}$ for Γ containing at most k intervals. The algorithm guesses in Step (1) the intervals in $\mathcal{S}_{apx}$ that are in $\mathcal{S}_{opt}$, and updates (Γ, k) accordingly. As explained before, we may assume that the intervals in the optimal solution sought (if it exists), $\mathcal{S}_{opt}$, are pairwise disjoint, start at segment endpoints, and that each interval in $\mathcal{S}_{opt}$ intersects two consecutive intervals in $\mathcal{S}_{apx}$. The algorithm then guesses which endpoints of intervals in $\mathcal{S}_{apx}$ are anchors (w.r.t. $\mathcal{S}_{opt}$).

The algorithm in Step (2) removes anchors from the set Υ of anchors, so that each anchor is the left endpoint of its interval in $\mathcal{S}_{apx}$. Note that after this restructuring, each interval in the sought solution $\mathcal{S}_{opt}$ contains exactly one anchor in Υ. Moreover, any two anchors are more than 1 unit apart, and hence, if $|\Upsilon| > k$, then the algorithm can safely reject the current guess, as it does in Step (2), since no solution $\mathcal{S}_{opt}$ of size k conforming to the current guess exists. It is clear that a solution $\mathcal{S}_{opt}$ to (Γ, k) exists if and only if there exists a guess of a set Υ of anchors satisfying the above conditions.

The algorithm then proceeds to determining how the intervals in the solution sought should be "anchored" (or placed) around their anchors in order to cover all segments in Γ. To do so, the algorithm considers the intervals in $\mathcal{S}_{apx}$ from left to right. For an interval I under consideration, the set of segments $\mathcal{S}$ whose left endpoints lie on I must be covered by $\mathcal{S}_{opt}$. The right endpoints of the segments in $\mathcal{S}$ lie on at most two intervals of $\mathcal{S}_{apx}$; we argue the more complicated case in which these endpoints lie on exactly two intervals, I_1, I_2, where I_2 is the successor of I_1 in $\mathcal{S}_{apx}$, as this case subsumes the other one. The set of segments $\mathcal{S}$ can be partitioned into $\mathcal{S}_1$ and $\mathcal{S}_2$, as explained in Step (3) of the algorithm, depending on which interval in I_1, I_2 the right endpoint of the segment in $\mathcal{S}$ lies on. Let the anchors $\mathfrak{a}, \mathfrak{b}, \mathfrak{u}, \mathfrak{s}, \mathfrak{t}$ be as defined in Step (3) of the algorithm. Observe that, since each of the anchors $\mathfrak{a}$ and $\mathfrak{u}$ covers a prefix (possibly empty) of the sequence of segments in $\mathcal{S}_1$ when ordered from left to right, one of the two anchors must dominate the other w.r.t. $\mathcal{S}_1$. Similarly, each of the anchors $\mathfrak{b}$ and $\mathfrak{s}$ covers a suffix (possibly empty) of the sequence of segments in $\mathcal{S}_1$ (when ordered from left to right), and hence one must dominate the other w.r.t. $\mathcal{S}_1$. With respect to $\mathcal{S}_2$, one of the two anchors $\mathfrak{a}$ and $\mathfrak{s}$ must dominate the other, and one of $\mathfrak{b}$ and $\mathfrak{t}$ must dominate the other. The algorithm guesses each of these domination relations, for each interval $I \in \mathcal{S}_{apx}$ and set of segments $\mathcal{S}$ whose left endpoints lie on I. If the algorithm guesses correctly, then for each segment in Γ, it assigns each of its two endpoints to an anchor such that the segment is covered by an interval inducing one of the anchors assigned to its endpoints. After guessing the domination relations among the anchors, the algorithm creates an instance $\mathcal{F}$ of 2-Sat that, for each segment $S \in \Gamma$ and each endpoint of $\mathcal{S}$, associates a Boolean variable whose value is true if and only if the endpoint of the segment is covered by the interval inducing the anchor assigned to the endpoint (based on the domination relation). The algorithm then adds enforcement clauses to $\mathcal{F}$ ensuring (the converse) that a satisfying assignment to $\mathcal{F}$ corresponds to an assignment of segment endpoints to anchors satisfying: (1) All endpoints assigned to the same anchor can be covered by a unit interval (E2); and (2) if an endpoint x of a segment that is assigned to an anchor $\mathfrak{h}$ is covered by the interval $I_{\mathfrak{h}}$ inducing $\mathfrak{h}$, then any segment endpoint assigned to $\mathfrak{h}$ that is between x and $\mathfrak{h}$ is also covered by $I_{\mathfrak{h}}$ (E1).

Given the above, it is not difficult to verify that the instance (Γ, k) is a YES-instance of Cont-Equal-SC if and only if there is a guess for the algorithm that yields a YES-instance $\mathcal{F}$ of 2-Sat, and hence, that the algorithm is correct. Next we analyze the running time of the algorithm.

First, observe that computing $\mathcal{S}_{apx}$ can be done in $\mathcal{O}(n \lg n)$ time, as this can be done by sorting the endpoints in Γ. Moreover, all processing steps for the intervals in $\mathcal{S}_{apx}$ and segments Γ can be carried out in time $\mathcal{O}(n \lg n)$. Therefore, we only need to analyze the size of the search tree needed to simulate the guesses performed by the algorithm. The algorithm performs guessing only in Step (1) and Step (3).

In Step (1), the algorithm guesses a number $k' \in \{0, \ldots, k\}$, and then it guesses a subset of k' intervals in $\mathcal{S}_{apx}$. The total number of branches needed to simulate these guesses is at most $\sum_{k'=0}^{k} \binom{3k}{k'}$. For each guess of k' intervals, the algorithm removes these intervals

from $\mathcal{S}_{apx}$, updates Γ and sets $k = k - k'$. Removing k' intervals from $\mathcal{S}_{apx}$ leaves $\mathcal{S}_{apx}$ with at most $3k - k'$ intervals. The algorithm then guesses which endpoints of the (remaining) intervals in $\mathcal{S}_{apx}$ are anchors. Since we can assume that the anchors in question are left endpoints of their intervals, guessing the anchors can be simulated by choosing a subset of $(k - k')$ endpoints, out of the at most $(3k - k')$ left endpoints of the (remaining) intervals in $\mathcal{S}_{apx}$. Therefore, the total number of branches needed to simulate all guesses in Step (1) is at most $\sum_{k'=0}^{k} \binom{3k}{k'} \cdot \binom{3k-k'}{k-k'}$.

In Step (3), the algorithm guesses the domination relations among anchors. At this point, the number of anchors (by Step (2)) is at most $k - k'$. In the guessing, each guess made is w.r.t. an interval $I \in \mathcal{S}_{apx}$ and the two anchors $\mathfrak{a}$ and $\mathfrak{b}$, as defined in Step (3). We will charge each guess to the two anchors that play the roles of $\mathfrak{a}$ and $\mathfrak{b}$ w.r.t. some interval $I \in \mathcal{S}_{apx}$. There are four guesses made, resulting in sixteen cases, and each of $\mathfrak{a}$ and $\mathfrak{b}$ is involved in the same number of guesses. Therefore, eight cases need to be distinguished with respect to each of $\mathfrak{a}$ and $\mathfrak{b}$. Since each of the at most k' anchors can play the role of $\mathfrak{a}$ once and of $\mathfrak{b}$ once, over all intervals in $\mathcal{S}_{apx}$ (note that a domination relation involving an empty anchor is determined, not guessed), it follows that the total number of cases that each anchor can be involved in is sixteen, which results in a total number of branches of at most $2^{4k'}$ over all anchors.

It follows that the size of the search tree needed to simulate all the guesses performed by the algorithm is $\sum_{k'=0}^{k} \binom{3k}{k'} \cdot \binom{3k-k'}{k-k'} \cdot 2^{4k'}$. Next, we upper bound this expression.

Applying the well-known upper bound $\binom{r}{s} \leq (e \cdot r/s)^s$ on the binomial coefficient in both binomial terms $\binom{3k}{k'}$ and $\binom{3k-k'}{k-k'}$, where e is the base of the natural logarithm, and simplifying, we obtain:

$$\sum_{k'=0}^{k} \binom{3k}{k'} \cdot \binom{3k-k'}{k-k'} \cdot 2^{4k'} \tag{1}$$

$$\leq (3e)^k \cdot 2^{4k} + (3e)^k \cdot \sum_{k'=0}^{k-1} 2^{4k'} \cdot (k/k')^{k'} \cdot ((k-k'/3)/(k-k'))^{k-k'} \tag{2}$$

$$\leq (3e)^k \cdot 2^{4k} + (3e)^k \cdot \sum_{k'=0}^{k-1} 2^{4k'} \cdot (k/k')^{k'} \cdot e^{2k'/3} \tag{3}$$

$$= (3e)^k \cdot 2^{4k} + (3e)^k \cdot \sum_{k'=0}^{k-1} ((2^4 \cdot e^{2/3} \cdot k)/k')^{k'} \tag{4}$$

$$\leq (3e)^k \cdot 2^{4k} + (3e)^k \cdot \mathcal{O}((2^4 \cdot e^{2/3})^k) = \mathcal{O}((2^4 \cdot 3 \cdot e^{5/3})^k). \tag{5}$$

In Inequality (2), we split the summation – a minor technicality – in order to avoid a denominator of 0 in the term $((k-k'/3)/(k-k'))^{k-k'}$, resulting from approximating $\binom{3k-k'}{k-k'}$, when $k' = k$. We obtain Inequality (3) from Inequality (2), by upper bounding the term $((k-k'/3)/(k-k'))^{k-k'}$ by $e^{2k'/3}$. This is done by rewriting the term $((k-k'/3)/(k-k'))^{k-k'}$ in the form $(1+1/x)^x$, and using the well-known inequality $(1+1/x)^x \leq e$, for all $x > 0$. We obtain Inequality (5) from Inequality (4) by showing that the function $((2^4 \cdot e^{2/3} \cdot k)/k')^{k'}$ is increasing in k', which then can be used to upper bound the summation $\sum_{k'=0}^{k}((2^4 \cdot e^{2/3} \cdot k)/k')^{k'}$ by $\mathcal{O}((2^4 \cdot 3 \cdot e^{2/3})^k)$. ◀

We can obtain FPT algorithms for Disc-Equal-SC, Disc-Equal-Exact-SC, and Cont-Equal-Exact-SC as well. The ideas leading to the FPT algorithm for Disc-Equal-SC are the same as those for Cont-Equal-SC, albeit the technical details become more complicated and the running time is significantly worse. The complications are mainly due to

the stipulation that the covering intervals cannot be arbitrarily chosen, and must be selected from the set $\mathcal{I}$, given as input. This makes it harder to obtain an approximate solution with the desired properties – and leads to a worse approximation ratio, and to restructure the anchors. In particular, we can no longer make the simplifying assumptions about the structure of the intervals in $\mathcal{S}_{apx}$ and $\mathcal{S}_{opt}$. The FPT results for Disc-Equal-Exact-SC and Cont-Equal-Exact-SC are byproducts of that for Disc-Equal-SC.

▶ **Theorem 12.** *Disc-Equal-SC, Disc-Equal-Exact-SC, Cont-Equal-Exact-SC can be solved in time* $\mathcal{O}(2^{30k} \cdot (|\Gamma| \lg |\Gamma| + |\mathcal{I}| \lg |\mathcal{I}|))$*, and hence are FPT.*

6 Conclusion

In this paper, we considered several variants of segment covering by unit intervals. We established the NP-hardness of the restrictions of these problems to instances in which all segments have the same length. In addition to its importance per se, this result strengthens and implies a number of NP-hardness results in the literature. We also presented parameterized complexity results for several of these problems, showing their W[1]-hardness for the general case, and presenting FPT algorithms for their restrictions to instances in which all segments have the same length. Our work gives rise to two open questions:

1. What is the parameterized complexity of Cont-Exact-SC and Disc-Exact-SC?
2. What is the complexity of the restriction of Cont-Exact-SC and Disc-Exact-SC to instances in which all segments have length at most 1 unit?

References

1 A. Acharyya, S. Nandy, S. Pandit, and S. Roy. Covering segments with unit squares. *Computational Geometry: Theory and Applications*, 79:1–13, 2019.

2 E. Arkin, A. Banik, P. Carmi, G. Citovsky, M. Katz, J. Mitchell, and M. Simakov. Choice is hard. In *ISAAC*, volume 9472 of *Lecture Notes in Computer Science*, pages 318–328. Springer, 2015.

3 E. Arkin, A. Banik, P. Carmi, G. Citovsky, M. Katz, J. Mitchell, and M. Simakov. Conflict-free covering. In *CCCG*, 2015.

4 E. Arkin, A. Banik, P. Carmi, G. Citovsky, M. Katz, J. Mitchell, and M. Simakov. Selecting and covering colored points. *Discrete Applied Mathematics*, 250:75–86, 2018.

5 P. Bonsma, T. Epping, and W. Hochstättler. Complexity results on restricted instances of a paint shop problem for words. *Discrete Applied Mathematics*, 154(9):1335–1343, 2006.

6 Y. Chen, J. Flum, and M. Grohe. Machine-based methods in parameterized complexity theory. *Theoretical Computer Science*, 339(2-3):167–199, 2005.

7 M. Claverol, E. Khramtcova, E. Papadopoulou, M. Saumell, and C. Seara. Stabbing circles for sets of segments in the plane. *Algorithmica*, 80(3):849–884, 2018.

8 T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

9 M. Cygan, F. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

10 R. Diestel. *Graph Theory, 4th Edition*. Springer, 2012.

11 R. Downey and M. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, Berlin, Heidelberg, 2013.

12 T. Erlebach and E. J. van Leeuwen. Approximating geometric coverage problems. In *SODA*, pages 1267–1276. SIAM, 2008.

13 T. Erlebach and E.J. van Leeuwen. PTAS for weighted set cover on unit squares. In *APPROX-RANDOM*, volume 6302 of *Lecture Notes in Computer Science*, pages 166–177. Springer, 2010.

14 M. R. Garey and D. S. Johnson. The rectilinear steiner tree problem in NP complete. *SIAM Journal of Applied Mathematics*, 32:826–834, 1977.

15 A. Gupta, S. Kale, V. Nagarajan, R. Saket, and B. Schieber. The approximability of the binary paintshop problem. In *APPROX-RANDOM*, volume 8096 of *Lecture Notes in Computer Science*, pages 205–217. Springer, 2013.

16 S. Hartung and R. Niedermeier. Incremental list coloring of graphs, parameterized by conservation. *Theoretical Computer Science*, 494:86–98, 2013.

17 G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996.

18 J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

19 K. Kobylkin. Stabbing line segments with disks: Complexity and approximation algorithms. In *AIST*, volume 10716 of *Lecture Notes in Computer Science*, pages 356–367. Springer, 2017.

20 S. Langerman and P. Morin. Covering things with things. *Discrete & Computational Geometry*, 33(4):717–729, 2005.

21 R. Madireddy and A. Mudgal. Stabbing line segments with disks and related problems. In *CCCG*, pages 201–207, 2016.

22 J. Wang, W. Li, and J. Chen. A parameterized algorithm for the hyperplane-cover problem. *Theoretical Computer Science*, 411(44-46):4005–4009, 2010.