

1

University of Warsaw

2

Faculty of Mathematics, Informatics and Mechanics

3

Katarzyna Kowalska

Student no. 371053

4

5

Approximation and Parametrized Algorithms for Segment Set Cover

6

Master's thesis

7

in COMPUTER SCIENCE

8

Supervisor:

dr Michał Pilipczuk

Instytut Informatyki

9

June 2020

10 **Supervisor's statement**

11 Hereby I confirm that the presented thesis was prepared under my supervision and
12 that it fulfils the requirements for the degree of Master of Computer Science.

13 Date

Supervisor's signature

14 **Author's statement**

15 Hereby I declare that the presented thesis was prepared by me and none of its contents
16 was obtained by means that are against the law.

17 The thesis has never before been a subject of any procedure of obtaining an academic
18 degree.

19 Moreover, I declare that the present version of the thesis is identical to the attached
20 electronic version.

21 Date

Author's signature

22

Abstract

23 The work presents a study of different geometric set cover problems. It mostly focuses on
24 segment set cover and its connection to the polygon set cover.

25

Keywords

26 set cover, geometric set cover, FPT, $W[1]$ -completeness, APX-completeness, PCP theorem,
27 NP-completeness

28

Thesis domain (Socrates-Erasmus subject area codes)

29 11.3 Informatyka

30

31

Subject classification

32 D. Software

33 D.127. Blabalgorithms

34 D.127.6. Numerical blabalysis

35

Tytuł pracy w języku polskim

36 Algorytmy parametryzowania i trudność aproksymacji problemu pokrywania zbiorów
37 odcinkami na płaszczyźnie

Contents

39	1. Introduction	5
40	2. Definitions	7
41	2.1. Geometric Set Cover	7
42	2.2. Approximation	7
43	2.3. δ -extensions	7
44	3. APX-completeness Geometric Set Cover	9
45	3.1. APX-completeness for segments parallel to axes	9
46	3.1.1. MAX-(3,3)-SAT and statement of reduction	9
47	3.1.2. Reduction	11
48	3.1.2.1. VARIABLE-gadget	11
49	3.1.2.2. OR-gadget	12
50	3.1.2.3. CLAUSE-gadget	14
51	3.1.2.4. Summary	16
52	3.1.2.5. Summary of construction	16
53	3.1.3. Construction lemmas and proof of Lemma 1	16
54	4. FPT for Geometric Set Cover for segments with δ-extensions	21
55	4.1. FPT for segments	21
56	4.1.1. Axis-parallel segments	21
57	4.1.2. Segments in arbitrary directions	22
58	4.2. FPT for weighted segments with δ -extensions	23
59	5. W[1]-completeness for weighted segments in 3 directions	27
60	5.1. W[1]-completeness for weighted segments in 3 directions	27
61	5.2. What is missing	30
62	6. Geometric Set Cover with lines	31
63	6.1. Lines parallel to one of the axis	31
64	6.2. FPT for arbitrary lines	31
65	6.3. APX-completeness for arbitrary lines	31
66	6.4. 2-approximation for arbitrary lines	32
67	6.5. Connection with general set cover	32
68	7. Geometric Set Cover with polygons	33
69	7.1. State of the art	33
70	8. Conclusions	35

Chapter 1

Introduction

The Set Cover problem is one of the most common NP-complete problems. [tutaj referencja]
We are given a family of sets and have to choose the smallest subfamily of these sets that cover
all their elements. This problem naturally extends to settings where we put different weights
on the sets and look for the subfamily of the minimal weight. This problem is NP-complete
even without weights and if we put restrictions on what the sets can be. One of such variants
is Vertex Cover problem, where sets have size 2 (they are edges in a graph).

In this work we focus on another such variant where the sets correspond to some geometric
shapes and only some points of the plane have to be covered. When these shapes are rectangles
with edges parallel to the axis, the problem can be proven to be W[1]-complete (solution of
size k cannot be found in $n^o(k)$ time), APX-complete (for sufficiently small $\epsilon > 0$, the problem
does not admit $1 + \epsilon$ -approximation scheme) [referencje].

Some of these settings are very easy. Set cover with lines parallel to one of the axis can
be solved in polynomial time.

There is a notion of δ -expansions, which loosen the restrictions on geometric set cover. We
allow the objects to cover the points after δ -expansion and compare the result to the original
setting. This way we can produce both FPT and EPTAS for the rectangle set cover with
 δ -extensions [referencje].

Our contribution. In this work, we prove that unweighted geometric set cover with seg-
ments is fixed parameter tractable (FPT).

Moreover, we show that geometric set cover with segments is APX-complete for unweighted
axis-parallel segments, even with $1/2$ -extensions. So the problem for very thin rectangles
also can't admit PTAS. Therefore, in the efficient polynomial-time approximation scheme
(EPTAS) for *fat polygons* by [Har-Peled and Lee, 2009], the assumption about polygons
being fat is necessary.

Finally, we show that geometric set cover with weighted segments in 3 directions is
W[1]-complete. However, geometric set cover with weighted segments is FPT if we allow
 δ -extension.

This result is especially interesting, since it's counter-intuitive that the unweighted setting
is FPT and the weighted setting is W[1]-complete. Most of such problems (like vertex cover
or [wiecej przykladow]) are equally hard in both weighted and unweighted settings.

Chapter 2

Definitions

2.1. Geometric Set Cover

In the geometric set cover problem we are given \mathcal{P} – a set of objects, which are connected subsets of the plane, \mathcal{C} – a set of points in the plane. The task is to choose $\mathcal{R} \subseteq \mathcal{P}$ such that every point in \mathcal{C} is inside some element from \mathcal{R} and $|\mathcal{R}|$ is minimized.

In the parametrized setting for a given k , we only look for a solution \mathcal{R} such that $|\mathcal{R}| \leq k$.

In the weighted setting, there is some given weight function $f : \mathcal{P} \rightarrow \mathbb{R}^+$, and we would like to find a solution \mathcal{R} that minimizes $\sum_{R \in \mathcal{R}} f(R)$.

2.2. Approximation

Let us recall some definitions related to optimization problems that are used in the following sections.

Definition 1. A **polynomial-time approximation scheme (PTAS)** for a minimization problem Π is a family of algorithms \mathcal{A}_ϵ for every $\epsilon > 0$ such that \mathcal{A}_ϵ takes an instance I of Π and in polynomial time finds a solution that is within a factor $(1 + \epsilon)$ of being optimal. That means the reported solution has weight at most $(1 + \epsilon)\text{opt}(I)$, where $\text{opt}(I)$ is the weight of an optimal solution for I .

Definition 2. A problem Π is **APX-hard** if assuming $P \neq NP$, there exists $\epsilon > 0$ such that there is no polynomial-time $(1 + \epsilon)$ -approximation algorithm for Π .

2.3. δ -extensions

TODO PLACEHOLDER for introductory text

δ -extensions is one of the modifications to a problem, that makes geometric set cover problem easier, it has been already used in literature (place some refrence here).

Definition 3 (δ -extensions for center-symmetric objects). For any $\delta > 0$ and a center-symmetric object L with centre of symmetry $S = (x_s, y_s)$, the **δ -extension** of L is the object $L^{+\delta} = \{(1 + \delta) \cdot (x - x_s, y - y_s) + (x_s, y_s) : (x, y) \in L\}$, that is, $L^{+\delta}$ is the image of L under homothety centered at S with scale $(1 + \delta)$

The geometric set cover problem with δ -extensions is a modified version of geometric set cover where:

- We need to cover all the points in \mathcal{C} with objects from $\{P^{+\delta} : P \in \mathcal{P}\}$ (which always include no fewer points than the objects before δ -extensions);

- We look for a solution that is no larger than the optimum solution for the original problem. Note that it does not need to be an optimal solution in the modified problem.

Formally, we have the following.

Definition 4 (Geometric set cover problem with δ -extensions). The geometric set cover problem with δ -extensions is the problem where for an input instance $I = (\mathcal{P}, \mathcal{C})$, the task is to output a solution $\mathcal{R} \subseteq \mathcal{P}$ such that the δ -extended set $\{R^{+\delta} : R \in \mathcal{R}\}$ covers \mathcal{C} and is no larger than the optimal solution for the problem without extensions, i.e. $|\mathcal{R}| \leq |\text{opt}(I)|$.

TODO: Some text

Definition 5 (Geometric set cover PTAS with δ -extensions). We define a PTAS for geometric set cover with δ -extensions as a family of algorithms $\{\mathcal{A}_{\delta, \epsilon}\}_{\delta, \epsilon > 0}$ that each takes as an input instance $I = (\mathcal{P}, \mathcal{C})$, and in polynomial-time outputs a solution $\mathcal{R} \subseteq \mathcal{P}$ such that the δ -extended set $\{R^{+\delta} : R \in \mathcal{R}\}$ covers \mathcal{C} and is within a $(1 + \epsilon)$ factor of the optimal solution for this problem without extensions, i.e. $(1 + \epsilon)|\mathcal{R}| \leq |\text{opt}(I)|$.

Chapter 3

APX-completeness Geometric Set Cover

3.1. APX-completeness for segments parallel to axes

In this section we analyze whether there exists PTAS for geometric set cover for rectangles. We show that we can restrict this problem to a very simple setting: segments parallel to axes and allow $(1/2)$ -extension, and the problem is still APX-hard. Note that segments are just degenerated rectangles with one side being very narrow.

Our results can be summarized in the following theorem and this section aims to prove it.

Theorem 1. *(axis-parallel segment set cover with $1/2$ -extension is APX-hard). Unweighted geometric set cover with axis-parallel segments in 2D (even with $1/2$ -extension) is APX-hard. That is, assuming $P \neq NP$, there does not exist a PTAS for this problem.*

Theorem 1 implies the following.

Corollary 1. *(rectangle set cover is APX-hard). Unweighted geometric set cover with rectangles (even with $1/2$ -extension) is APX-hard.*

We prove Theorem 1 by taking a problem that is APX-hard and showing a reduction. For this problem we choose MAX-(3,3)-SAT which we define below.

3.1.1. MAX-(3,3)-SAT and statement of reduction

Definition 6. MAX-3SAT is the following maximization problem. We are given a 3-CNF formula, and need to find an assignment of variables that satisfies the most clauses.

Definition 7. MAX-(3,3)-SAT is a variant of MAX-3SAT with an additional restriction that every variable appears in exactly 3 clauses. Note that thus, the number of clauses is equal to the number of variables.

In our proof of Theorem 1 we use hardness of approximation of MAX-(3,3)-SAT proved in [Håstad, 2001] and described in Theorem 2 below.

Definition 8 (α -satisfiable MAX-3SAT formula). MAX-3SAT formula of size n is at most α -satisfiable, if every assignment of variables satisfies no more than αn clauses.

Theorem 2. [Håstad, 2001]

For any $\epsilon > 0$, it is NP-hard to distinguish satisfiable (3,3)-SAT formulas from at most $(7/8 + \epsilon)$ -satisfiable (3,3)-SAT formulas.

Given an instance I of MAX-(3,3)-SAT, we construct an instance J of axis-parallel segment set cover problem, such that for a sufficiently small $\epsilon > 0$, a polynomial time $(1 + \epsilon)$ -approximation algorithm for J would be able to distinguish whether an instance I of MAX-(3,3)-SAT is fully satisfiable or is at most $(7/8 + \epsilon)$ -satisfiable. However, according to (Theorem 2) the latter problem is NP-hard. This would imply $P = NP$, contradicting the assumption.

The following lemma encapsulates the properties of the reduction described in this section, and it allows us to prove Theorem 1.

Lemma 1. *Given an instance S of MAX-(3,3)-SAT with n variables and optimum value $opt(S)$, we can construct an instance I of geometric set cover with axis-parallel segments in $2D$, such that:*

(1) *For every solution X of instance I , there exists a solution of S that satisfies at least $15n - |X|$ clauses.*

(2) *For every solution of instance S that satisfies w clauses, there exists a solution of I of size $15n - w$.*

(3) *Every solution with $1/2$ -extensions of I is also a solution to the original instance I .*

Therefore, the optimum size of a solution of I is $opt(I) = 15n - opt(S)$.

We prove Lemma 1 in subsequent sections, but meanwhile let us prove Theorem 1 using Lemma 1 and Theorem 2.

Proof of Theorem 1.

Consider any $0 < \epsilon < 1/(15 \cdot 8)$.

Let us assume that there exists a polynomial-time $(1 + \epsilon)$ -approximation algorithm for unweighted geometric set cover with axis-parallel segments in $2D$ with $(1/2)$ -extensions. We construct an algorithm that solves the problem stated in Theorem 2, thereby proving that $P = NP$.

Take an instance S of MAX-(3,3)-SAT to be distinguished and construct an instance of geometric set cover I using Lemma 1. We now use the $(1 + \epsilon)$ -approximation algorithm for geometric set cover on I . Denote the size of the solution returned by this algorithm as $approx(I)$. We prove that if in S one can satisfy at most $(\frac{7}{8} + \epsilon)n$ clauses, then $approx(I) \geq 15n - (\frac{7}{8} + \epsilon)n$ and if S is satisfiable, then $approx(I) < 15n - (\frac{7}{8} + \epsilon)n$.

Assume S satisfiable. From the definition of S being satisfiable, we have:

$$opt(S) = n.$$

From Lemma 1 we have:

$$opt(I) = 14n.$$

Therefore,

$$\begin{aligned} approx(I) &\leq (1 + \epsilon)opt(I) = 14n(1 + \epsilon) = 14n + 14\epsilon \cdot n = \\ &= 14n + (15\epsilon - \epsilon)n < 14n + \left(\frac{1}{8} - \epsilon\right)n = 15n - \left(\frac{7}{8} + \epsilon\right)n \end{aligned}$$

Assume S is at most $(\frac{7}{8} + \epsilon)$ satisfiable. From the definition of S being at most $(\frac{7}{8} + \epsilon)n$ satisfiable, we have:

$$opt(S) \leq \left(\frac{7}{8} + \epsilon\right)n$$

From Lemma 1 we have:

$$opt(I) \geq 15n - \left(\frac{7}{8} + \epsilon\right)n$$

207 Since a solution to I with $\frac{1}{2}$ -extensions is also a solution without extensions, by Lemma 1
208 (3.), we have:

$$approx(I) \geq opt(I) = 15n - \left(\frac{7}{8} + \epsilon\right)n$$

209 Therefore, by using the assumed $(1 + \epsilon)$ -approximation algorithm, it is possible to dis-
210 tinguish the case when S is satisfiable from the case when it is at most $(\frac{7}{8} + \epsilon)n$ satisfiable,
211 it suffices to compute $approx(I)$ with $15n - (\frac{7}{8} + \epsilon)n$. Hence, the assumed approximation
212 algorithm cannot exist, unless $P = NP$. \square

213 3.1.2. Reduction

214 We proceed to the proof of Lemma 1. That is, we show a reduction from MAX-(3,3)-SAT
215 problem to geometric set cover with segments parallel to axis. Moreover, the obtained instance
216 of geometric set cover will be robust to $1/2$ -extensions (have the same optimal solution after
217 $1/2$ -extension).

218 The construction will be composed of 2 types of gadgets: **VARIABLE-gadgets** and
219 **CLAUSE-gadgets**. **CLAUSE-gadgets** would be constructed using two **OR-gadgets** con-
220 nected together.

221 3.1.2.1. VARIABLE-gadget

222 **VARIABLE-gadget** is responsible for choosing the value of a variable in a CNF formula. It
223 allows two minimum solutions of size 3 each. These two choices correspond to the two Boolean
224 values of the variable corresponding to this gadget.

225 **Points.** Define points a, b, c, d, e, f, g, h as follows, where $L = 12n$:

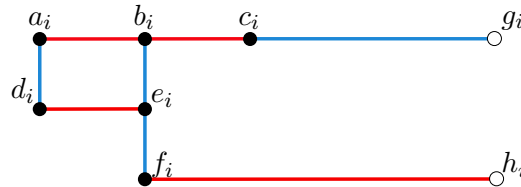


Figure 3.1: **VARIABLE-gadget**. We denote the set of points marked with black circles as pointsVariable_i , and they need to be covered (are part of the set \mathcal{C}). Note that some of the points are not marked as black dots and exists only to name segments for further reference. We denote the set of red segments as $\text{chooseVariable}_i^{\text{false}}$ and the set of blue segments as $\text{chooseVariable}_i^{\text{true}}$.

$$\begin{array}{llll}
a = (-L, 0) & b = (-\frac{2}{3}L, 0) & c = (-\frac{1}{3}L, 0) & d = (-L, 1) \\
e = (-\frac{2}{3}L, 1) & f = (-\frac{2}{3}L, 2) & g = (L, 0) & h = (L, 2)
\end{array}$$

Let us define:

$$\text{pointsVariable} = \{a, b, c, d, e, f\}$$

and

$$\text{pointsVariable}_i = \text{pointsVariable} + (0, 4i)$$

We denote $a_i = a + (0, 4i)$ etc.

Segments. Let us define:

$$\text{chooseVariable}_i^{\text{true}} = \{(a_i, d_i), (b_i, f_i), (c_i, g_i)\}$$

$$\text{chooseVariable}_i^{\text{false}} = \{(a_i, c_i), (d_i, e_i), (f_i, h_i)\}$$

$$\text{segmentsVariable}_i = \text{chooseVariable}_i^{\text{true}} \cup \text{chooseVariable}_i^{\text{false}}$$

Lemma 2. For any $1 \leq i \leq n$, points in pointsVariable_i can be covered using 3 segments from $\text{segmentsVariable}_i$.

Proof. We can use either set $\text{chooseVariable}_i^{\text{true}}$ or $\text{chooseVariable}_i^{\text{false}}$. \square

Lemma 3. For any $1 \leq i \leq n$, points in pointsVariable_i can not be covered with fewer than 3 segments from $\text{segmentsVariable}_i$.

Proof. No segment of $\text{segmentsVariable}_i$ covers more than one point from $\{d_i, f_i, c_i\}$, therefore pointsVariable_i can not be covered with fewer than 3 segments. \square

Lemma 4. For every set $A \subseteq \text{segmentsVariable}_i$ such that A covers pointsVariable_i and $(c_i, g_i), (f_i, h_i) \in A$, it holds that $|A| \geq 4$.

Proof. No segment from $\text{segmentsVariable}_i$ covers more than one point from $\{a_i, e_i\}$, therefore $\text{pointsVariable}_i - \{c_i, f_i, g_i, h_i\}$ can not be covered with fewer than 2 segments. \square

3.1.2.2. OR-gadget

OR-segment connects input and output segments that are connected to other parts of constructions.

Output segment is part of OR-segment, but input is not.

For every solution \mathcal{R} of the whole construction. Define \mathcal{R}' as intersection of \mathcal{R} and the gadget segments. Minimum solution of OR-gadget has size w , i.e. $|\mathcal{R}'| \leq w$. *output* segments can be part of \mathcal{R}' only if *input_x* or *input_y* are part of the chosen solution \mathcal{R} . If none of them are chosen, then solution containing *output* segment has weight at least $w + 1$. Therefore the following formula holds:

$$\text{output} \in \mathcal{R}' \wedge |\mathcal{R}'| = w \Rightarrow (x \in \mathcal{R}) \vee (y \in \mathcal{R})$$

Only 3 points that belong to this segment: $l_{i,j}, p_{i,j}, v_{i,j}$ can be covered by segment not from the OR-gadget.



Figure 3.2: **OR-gadget**. Figure presenting OR-gadget: segments from $\text{chooseOr}_{i,j}^{false}$ are red, segments from $\text{chooseOr}_{i,j}^{true}$ are blue, segments from $\text{orMoveVariable}_{i,j}$ are yellow and green. Dark blue segment is an *output* segment. Grey segments $input_x$ and $input_y$ are input segments that are not part of $\text{segmentsOr}_{i,j}$.

251 **Points.**

$$\begin{array}{llll}
 l_0 = (0, 0) & m_0 = (0, 1) & n_0 = (0, 2) & o_0 = (0, 3) \\
 p_0 = (0, 4) & q_0 = (1, 1) & r_0 = (1, 3) & s_0 = (2, 1) \\
 t_0 = (2, 2) & u_0 = (2, 3) & v_0 = (3, 2) &
 \end{array}$$

$$vec_{i,j} = (10i + 3 + 3j, 4n + 2j)$$

253 Define $\{l_{i,j}, m_{i,j} \dots v_{i,j}\}$ as $\{l_0, m_0 \dots v_0\}$ shifted by $vec_{i,j}$

254 Note that $v_{i,0} = l_{i,1}$ (see Figure 3.3)

$$\text{pointsOr}_{i,j} = \{l_{i,j}, m_{i,j}, n_{i,j}, o_{i,j}, p_{i,j}, q_{i,j}, r_{i,j}, s_{i,j}, t_{i,j}, u_{i,j}\}$$

255 Note that $\text{pointsOr}_{i,j}$ does not include point $v_{i,j}$

256 **Segments.** We define names subsets of segments, to refer to them in lemmas.

$$\text{chooseOr}_{i,j}^{false} = \{(q_{i,j}, r_{i,j}), (s_{i,j}, u_{i,j})\}$$

$$\text{chooseOr}_{i,j}^{true} = \{(m_{i,j}, s_{i,j}), (o_{i,j}, u_{i,j}), (t_{i,j}, v_{i,j})\}$$

$$\text{orMoveVariable}_{i,j} = \{(l_{i,j}, n_{i,j}), (n_{i,j}, p_{i,j})\}$$

257 Segments in OR-gadget:

$$\text{segmentsOr}_{i,j} = \text{chooseOr}_{i,j}^{false} \cup \text{chooseOr}_{i,j}^{true} \cup \text{orMoveVariable}_{i,j}$$

258 **Lemma 5.** For any $1 \leq i \leq n, j \in \{0, 1\}$ and $x \in \{l_{i,j}, p_{i,j}\}$, points in $\text{pointsOr}_{i,j} - \{x\} \cup \{v_{i,j}\}$
 259 can be covered with 4 segments from $\text{segmentsOr}_{i,j}$.

260 *Proof.* We can do that using one segment from $\text{orMoveVariable}_{i,j}$, the one that does not cover
 261 x , and all segments from $\text{chooseOr}_{i,j}^{true}$. \square

262 **Lemma 6.** For any $1 \leq i \leq n, j \in \{0, 1\}$, points in $\text{pointsOr}_{i,j}$ can be covered with 4 segments
 263 from $\text{segmentsOr}_{i,j}$.

264 *Proof.* We can do that using segments from $\text{orMoveVariable}_{i,j}$ and $\text{chooseOr}_{i,j}^{\text{false}}$. \square

265 3.1.2.3. CLAUSE-gadget

266 CLAUSE-gadget is responsible for calculating if variables values assigned in variable gadgets
 267 satisfy the respective clause in CNF. It has minimum solution of weight w if and only if the
 268 clause is satisfied, i.e. at least one of the respective variables is assigned a correct value.
 269 Otherwise it has minimum solution of weight $w + 1$. This way, by analyzing the minimum
 270 solution for the whole problem, we can tell how many clauses were possible to satisfy in the
 271 optimum solution of CNF.

272 The CLAUSE-gadgets consist of two OR-gadgets. It would be inconvenient to posi-
 273 tion the CLAUSE-gadgets in between the very long variable segments. Instead, we use
 274 a simple auxiliary gadget to *transfer* whether the segment is in a solution, i.e. segments
 275 $(x_{i,0}, x_{i,1}), (y_{i,0}, y_{i,1}), (z_{i,0}, z_{i,1})$. Each gadget consists of two segments $(x_{i,0}, x_{i,1}), (x_{i,1}, a)$.
 276 These are the only segments that can cover $x_{i,1}$. If $x_{i,0}$ is already covered by some other
 277 gadget, we can cover $x_{i,1}$ by the other segment covering another point from the gadget, say a .
 278 If $x_{i,0}$ is not covered, then the only way to cover $x_{i,0}$ is to use segment $(x_{i,0}, x_{i,1})$. Intuitively,
 279 the two segments *transfer* the state of $x_{i,0}$ onto a , but there are less restrictions on where a
 280 can be placed, simplifying the construction.



Figure 3.3: **CLAUSE-gadget.** This figure presents CLAUSE-gadget. Every green rectangle is an OR-gadget. y -coordinates of $x_{i,0}$, $y_{i,0}$ and $z_{i,0}$ depend on the variables in the i -th clause. Grey segments corresponds to the values of variables satisfying the i -th clause.

281 **Points.** TODO: Rephrase it

282 Assuming clause $C_i = a \vee b \vee c$, function $\text{idx}(w)$ returns index of the variable w , function
 283 $\text{neg}(w)$ returns whether variable w is negated in a clause.

$$\begin{aligned}
x_{i,0} &= (10i + 1, 4 \cdot idx(a) + 2 \cdot neg(c)) & x_{i,1} &= (10i + 1, 4n) \\
y_{i,0} &= (10i + 2, 4 \cdot idx(b) + 2 \cdot neg(b)) & y_{i,1} &= (10i + 2, 4n + 4) \\
z_{i,0} &= (10i + 3, 4 \cdot idx(c) + 2 \cdot neg(c)) & z_{i,1} &= (10i + 3, 4n + 6)
\end{aligned}$$

$$\text{moveVariable}_i = \{x_{i,j} : j \in \{0, 1\}\} \cup \{y_{i,j} : j \in \{0, 1\}\} \cup \{z_{i,j} : j \in \{0, 1\}\}$$

$$\text{pointsClause}_i = \text{moveVariable}_i \cup \text{pointsOr}_{i,0} \cup \text{pointsOr}_{i,1} \cup \{v_{i,1}\}$$

Segments.

$$\begin{aligned}
\text{segmentsClause}_i &= \{(x_{i,0}, x_{i,1}), (y_{i,0}, y_{i,1}), (z_{i,0}, z_{i,1}), (x_{i,1}, l_{i,0}), (y_{i,1}, p_{i,0}), (z_{i,1}, p_{i,1}), \} \cup \\
&\cup \text{segmentsOr}_{i,0} \cup \text{segmentsOr}_{i,1}
\end{aligned}$$

Lemma 7. For any $1 \leq i \leq n$ and $a \in \{x_{i,0}, y_{i,0}, z_{i,0}\}$, there is a $\text{solClause}_i^{\text{true},a} \subset \text{segmentsClause}_i$ with $|\text{solClause}_i^{\text{true},a}| = 11$ that covers points in $\text{pointsClause}_i - \{a\}$.

Proof. For $a = x_{i,0}$ (analogous proof for $y_{i,0}$): First we use Lemma 5 twice with excluded $x = l_{i,0}$ and $x = l_{i,1} = v_{i,0}$, resulting with 8 segments $\text{chooseOr}_{i,0}^{\text{true}} \cup \text{chooseOr}_{i,1}^{\text{true}}$ which cover all required points apart from $x_{i,1}, y_{i,0}, y_{i,1}, z_{i,0}, z_{i,1}, l_{i,0}$. We cover those using additional 3 segments: $\{(x_{i,1}, l_{i,0}), (y_{i,0}, y_{i,1}), (z_{i,0}, z_{i,1})\}$

For $a = z_{i,0}$: Using Lemma 6 and Lemma 5 with $x = p_{i,1}$, resulting with 8 segments $\text{chooseOr}_{i,0}^{\text{false}} \cup \text{chooseOr}_{i,1}^{\text{true}}$ which cover all required points apart from $x_{i,0}, x_{i,1}, y_{i,0}, y_{i,1}, z_{i,1}, p_{i,1}$. We cover those using additional 3 segments: $\{(x_{i,0}, x_{i,1}), (y_{i,0}, y_{i,1}), (z_{i,1}, p_{i,1})\}$. \square

Lemma 8. For any $1 \leq i \leq n$ there is $\text{solClause}_i^{\text{false}} \subset \text{segmentsClause}_i$ with $|\text{solClause}_i^{\text{false}}| = 12$ that covers points in pointsClause_i .

Proof. Using Lemma 6 twice we can cover $\text{pointsOr}_{i,0}$ and $\text{pointsOr}_{i,1}$ with 8 segments.

To cover the remaining points we additionally use: $\{(x_{i,0}, x_{i,1}), (y_{i,0}, y_{i,1}), (z_{i,0}, z_{i,1}), (t_{i,1}, v_{i,1})\}$ \square

Lemma 9. For any $1 \leq i \leq n$:

(1) points in pointsClause_i can not be covered using any subset of segments from segmentsClause_i of size smaller than 12;

(2) points in $\text{pointsClause}_i - \{x_{i,0}, y_{i,0}, z_{i,0}\}$ can not be covered using any subset of segments from segmentsClause_i of size smaller than 11.

Proof of (1). No segment in segmentsClause_i covers more than 2 points from $\{x_{i,0}, y_{i,0}, z_{i,0}, l_{i,0}, p_{i,0}, q_{i,0}, u_{i,0}, v_{i,0}, l_{i,1}, p_{i,1}, q_{i,1}, u_{i,1}, v_{i,1}\}$.

Therefore we need to use at least 12 segments. \square

Proof of (2). We can choose disjoint sets X, Y, Z such that $X \cup Y \cup Z \subseteq \text{pointsClause}_i - \{x_{i,0}, y_{i,0}, z_{i,0}\}$ and there are no segments covering points from different sets. And we prove lower bounds for each of these sets.

$$X = \{x_{i,1}, y_{i,1}, z_{i,1}\}$$

No two points in X are covered with one segment of segmentsClause_i , so it must be covered with 3 different segments.

$$Y = \text{pointsOr}_{i,0} - \{l_{i,0}, p_{i,0}\}$$

$$Z = \text{pointsOr}_{i,1} - \{l_{i,1}, p_{i,1}\}$$

For both Y and Z we can check all of the subsets of 3 segments of segmentsClause_i with brutforce that none of them cover the set of points, so both Y and Z have to be covered with disjointed sets of 4 segments.

TODO: Funny fact, neither Y nor Z doesn't have independent set of size 4.

Therefore pointsClause_i must be covered with at least $3 + 4 + 4 = 11$ segments. \square

3.1.2.4. Summary

Add some smart lemmas that sets will be exclusive to each other.

Lemma 10. Robustness to 1/2-extensions. *For every segment $s \in \mathcal{P}$, s and $s^{+1/2}$ cover the same points from \mathcal{C} .*

Proof. We can just check every segment. Most of the segments s are collinear only with points that lay on s , so trivially $s^{+1/2}$ cannot cover more points than s does.

TODO: list problematic segments here

In the same gadget: $(n_{i,j}, p_{i,j})$ does not cover $m_{i,j}$ and symmetrically. $(t_{i,j}, v_{i,j})$ does not cover $n_{i,j}$. $(o_{i,0}, u_{i,0})$ does not cover $m_{i,1}$ and symmetrically. $(y_{i,1}, p_{i,0})$ does not cover $n_{i,j}$.

From different gadgets: (b_i, f_i) after $\frac{1}{2}$ -extensions does not cover b_{i+1} point.

VARIABLE-gadget's (a_i, c_i) after $\frac{1}{2}$ -extensions does not cover any points $x_{i,0}, y_{i,0}$ or $z_{i,0}$ from CLAUSE-gadget.

\square

3.1.2.5. Summary of construction

We define:

$$\mathcal{C} := \bigcup_{1 \leq i \leq n} \text{pointsVariable}_i \cup \text{pointsClause}_i$$

$$\mathcal{P} := \bigcup_{1 \leq i \leq n} \text{segmentsVariable}_i \cup \text{segmentsClause}_i$$

The subsequent sections define these sets.

We prove some properties of different gadgets. Every segment for a gadget will only cover points in this gadget (won't interact with any different gadget), so we can prove lemmas *locally*.

TODO: y axis is increasing values downward on figures (not upwards like in normal).

3.1.3. Construction lemmas and proof of Lemma 1

In order to prove Lemma 1 we introduce several auxiliary lemmas proving properties of the construction described in the previous section.

Consider an instance S of MAX-(3,3)-SAT of size n with optimum solution satisfying k clauses. Let us construct an instance $(\mathcal{C}, \mathcal{P})$ of geometric set cover as described in Section 3.1.2 for instance S of MAX-(3,3)-SAT.

Lemma 11. *Instance $(\mathcal{C}, \mathcal{P})$ of geometric set cover admits a solution of size $15n - k$.*



Figure 3.4: **General schema.**

General layout of VARIABLE-gadget and CLAUSE-gadget and how they interact with each other.

TODO: Rename Choose X to VARIABLE-gadget and Clause C to CLAUSE-gadget.

Proof. Let the clauses in S be $c_1, c_2 \dots c_n$ and the variables be $x_1, x_2 \dots x_n$. Let the assignment of the variables in the optimum solution to S be $\phi : \{x_1, x_2 \dots x_n\} \rightarrow \{\text{true}, \text{false}\}$.

We cover every VARIABLE-gadget with solution described in Lemma 2, in the i -th gadget choosing the set of segments corresponding to the value of $\phi(x_i)$.

For every clause that is satisfied, say c_i , let us name the variable that is **true** in it as x_i and point corresponding to x_i in pointsClause_i as a . Points in pointsClause_i are covered with set $\text{solClause}_i^{\text{true}, a}$ described in Lemma 7. For every clause that is not satisfied, say c_j , points in pointsClause_j are covered with set $\text{solClause}_j^{\text{false}}$ described in Lemma 8.

Formally we define sets responsible for choosing variable and satisfying the variable, R_i and C_i respectively, as following:

$$\begin{aligned} R_i &= \begin{cases} \text{chooseVariable}_i^{\text{true}} & \text{if } \phi(x_i) = \text{true} \\ \text{chooseVariable}_i^{\text{false}} & \text{if } \phi(x_i) = \text{false} \end{cases} \\ C_i &= \begin{cases} \text{solClause}_i^{\text{true}, a} & \text{if } c_i \text{ satisfied} \\ \text{solClause}_i^{\text{false}} & \text{if } c_i \text{ not satisfied} \end{cases} \\ \mathcal{R} &= \bigcup_{i=1}^n \{R_i \cup C_i : 1 \leq i \leq n\}. \end{aligned}$$

This set covers all the points from \mathcal{C} , because the sets R_i, C_i individually cover their corresponding gadgets, as proved in the respective lemmas.

All of these sets are disjoint, so the size of the obtained solution is:

$$|\mathcal{R}| = \sum_{i=1}^n R_i + \sum_{i=1}^n C_i = 3n + 11k + 12(n - k) = 15n - k.$$

□

Lemma 12. Suppose we have a solution \mathcal{R} of the instance $(\mathcal{C}, \mathcal{P})$ of geometric set cover. Then there exists a solution \mathcal{R}' , such that $|\mathcal{R}'| \leq |\mathcal{R}|$, and for each VARIABLE-gadget \mathcal{R}' contains at most one of the segments (c_i, g_i) and (f_i, h_i) .

Proof. Assume that we have $\{(c_i, g_i), (f_i, h_i)\} \subseteq \mathcal{R}$ for some i . We will show how to modify \mathcal{R} into \mathcal{R}' , such that the number of such i decreases, while \mathcal{R}' is still a valid solution of $(\mathcal{C}, \mathcal{P})$, and $|\mathcal{R}'| \leq |\mathcal{R}|$. Then, by repeating this procedure, we can eventually construct a solution satisfying the property from the Lemma.

To construct \mathcal{R}' , we remove either (c_i, g_i) or (f_i, h_i) from \mathcal{R} , and then add one extra segment to make \mathcal{R}' valid. Recall that the i -th VARIABLE-gadget corresponds to variable x_i in S . As every variable in S is used in exactly 3 clauses, one of the ways of setting x_i (to either **true** or **false**) must satisfy at least 2 clauses. If that setting is $x_i = \text{true}$, then we remove (f_i, h_i) , otherwise we remove (c_i, g_i) . Now, there exists at most one CLAUSE-gadget which needs adjustment to make \mathcal{R}' valid; we do that by adding $(t_{j,1}, v_{j,1})$ to \mathcal{R}' .

TODO: Can we really just remove one segment and add another one? I'd think we need to "restructure" \mathcal{R} around pointsVariable_i (saving one segment due to Lemma 3 and Lemma 4) and then again restructure \mathcal{R} around the clause that we need to fix? □

Lemma 13. Suppose we have a solution \mathcal{R} of the instance $(\mathcal{C}, \mathcal{P})$ of geometric set cover that is of size w . Then there exists a solution of S that satisfies at least $15n - w$ clauses.

374 *Proof.* Let the clauses in S be $c_1, c_2 \dots c_n$ and the variables be $x_1, x_2 \dots x_n$. Given a solution
 375 \mathcal{R} of the instance $(\mathcal{C}, \mathcal{P})$ of geometric set cover, we use Lemma 12 to modify \mathcal{R} such that for
 376 any i it contains at most one of (c_i, g_i) and (f_i, h_i) ; this may decrease the cost of \mathcal{R} , but that
 377 does not matter in the subsequent construction. To simplify notation, in the remainder of
 378 this proof we use \mathcal{R} to refer to the modified solution.

379 Given \mathcal{R} , we construct a solution of S by constructing an assignment of variables $\phi :$
 380 $\{x_1, x_2 \dots x_n\} \rightarrow \{\text{true}, \text{false}\}$ that satisfies at least $15n - w$ clauses in S .

381 **Variables** Recall that due to Lemma 12, \mathcal{R} contains at most one of (c_i, g_i) and (f_i, h_i) .
 We define the value $\phi(x_i)$ for the variable x_i as follows:

$$\begin{cases} \phi(x_i) = \text{true} & \text{if } (c_i, g_i) \in \mathcal{R} \\ \phi(x_i) = \text{false} & \text{if } (f_i, h_i) \in \mathcal{R} \\ \phi(x_i) = \text{false} & \text{otherwise} \end{cases} \quad (3.1)$$

382 Moreover, from Lemma 3 we get $|\text{pointsVariable}_i \cap \mathcal{R}| \geq 3$ for every i .

383 **Clauses** For a clause $C_i = x \vee y \vee z$, \mathcal{R} needs to use at least 11 segments to cover
 384 $\text{pointsClause}_i - \{x, y, z\}$ in CLAUSE-gadget (Lemma 9).

385 TODO: maybe put something with cases and names of sets as above

386 Moreover, if all of the points $\{x_{i,0}, y_{i,0}, z_{i,0}\}$ are not covered by the segments from $\mathcal{R} \cap \text{pointsVariable}_i$,
 387 then \mathcal{R} needs to cover pointsClause_i with at least 12 segments by Lemma 9.

TODO: Maybe remove section below, because we do this calculation at the end anyway
 We covered CLAUSE-gadget with at least 11 or at least 12 segments:

$$|\bigcup_{i=1}^n \text{segmentsClause}_i \cap \mathcal{R}| \geq 11n + a$$

388 where a is the number of clauses where none of the points $x_{i,0}, y_{i,0}, z_{i,0}$ were covered by
 389 $\mathcal{R} \cap \text{segmentsVariable}_j$ for their respective variable x_j .

390 **Satisfied clauses with chosen variable assignment.** Consider a clause, say c_i . If
 391 none of the points $x_{i,0}, y_{i,0}, z_{i,0}$ in pointsClause_i were covered by segments from $\mathcal{R} \cap \text{segmentsVariable}_j$,
 392 this clause is not satisfied by assignment ϕ .

393 If one of these points is covered by segments from VARIBALE-gadget (TODO better this
 394 or $\mathcal{R} \cap \text{segmentsVariable}_j$), then denote this point as t and say it corresponds to variable x_j .
 395 Consider the cases of choosing value of $\phi(x_j)$ in equation (3.1).

396 If \mathcal{R} contains exactly one of the segments (c_j, g_j) and (f_j, h_j) , then the value $\phi(x_j)$ satisfies
 397 c_i .

398 If \mathcal{R} contains neither (c_j, g_j) nor (f_j, h_j) , then it is impossible that t is covered by segments
 399 in $\mathcal{R} \cap \text{segmentsVariable}_j$.

400 This means that ϕ satisfies all but at most a clauses in S .

401 To conclude, we proved that given a solution of $(\mathcal{C}, \mathcal{P})$ of size w , we have constructed a
 402 variables assignment ϕ that satisfies at least $n - a$ clauses of S . Finally, note that

$$w \geq 3n + 11(n - a) + 12a = 3n + 11n + a = 14n + a,$$

hence

$$15n - w \leq 15n - 14n - a = n - a.$$

403 So ϕ satisfies at least $15n - w$ clauses of S . □

404 We are ready to conclude the proof of Lemma 1.

Proof of Lemma 1. By Lemma 11, we know that there exists a solution to $(\mathcal{C}, \mathcal{P})$ of size $15n - k$, so:

$$opt((\mathcal{C}, \mathcal{P})) \leq 15n - k.$$

Since the optimum solution of S satisfies k clauses, then according to Lemma 13:

$$opt((\mathcal{C}, \mathcal{P})) \geq 15n - k.$$

405 Therefore, the solution given by Lemma 11 of size $15n - k$ is an optimum solution to the
406 instance $(\mathcal{C}, \mathcal{P})$. □

Chapter 4

FPT for Geometric Set Cover for segments with δ -extensions

4.1. FPT for segments

In this section we consider the fixed-parameter tractable algorithms for unweighted geometric set cover with segments. Setting where segments are limited to be axis-parallel (or limited to constant number of directions) has an FPT algorithm already present in literature. We present an FPT algorithm for unweighted geometric set cover with segments, where segments are in arbitrary directions.

4.1.1. Axis-parallel segments

You can find this in Platypus book. (TODO add referece)

We show an $\mathcal{O}(2^k)$ -time branching algorithm. In each step, the algorithm selects a point a which is not yet covered, branches to choose one of the two directions, and greedily chooses a segment in that direction to cover a . This proceeds until either all points are covered or k segments are chosen.

Let us take the point $a = (x_a, y_a)$ which is the smallest among points that are not yet covered in the lexicographic ordering of points in \mathbb{R}^2 . We need to cover a with some of the remaining segments.

Branch over the choice of one of the coordinates (x or y); without loss of generality, let us assume we chose x . Among the segments lying on line $x = x_a$, we greedily add to the solution the one that covers the most points. As a was the smallest in the lexicographical order, then all points on line $x = x_a$ have the y -coordinate larger than y_a . Therefore, if we denote the greedily chosen segment as s , then any other segment on $x = x_a$ that covers a can only cover a (possibly improper) subset of points covered by s . Thus, greedily choosing s is optimal.

In each step of the algorithm we add one segment to the solution, thus each branch can stop at depth k . If no branch finds a solution, then that means a solution of size at most k does not exist.

TODO: Maybe split it into theorem + algorithm + explanation like in section 4.1.2

Remark 1. *The same algorithm can be used for segments in d directions, where we branch over d directions and it runs in complexity $\mathcal{O}(d^k)$.*

4.1.2. Segments in arbitrary directions

In this section we consider setting where segments are not constrained to only d directions. We present a fixed-parameter tractable algorithm, where parameter is the size of the solution.

Theorem 3. (FPT for segment cover). *There exists an algorithm that given a family \mathcal{P} of n segments (in any direction), a set of m points \mathcal{C} and a parameter k , runs in time $k^{O(k)} \cdot (nm)^2$, and outputs a subfamily $\mathcal{R} \subseteq \mathcal{P}$ such that $|\mathcal{R}| \leq k$ and \mathcal{R} covers all points in \mathcal{C} , or determines that such a set \mathcal{R} does not exist.*

We will need the following lemmas.

Lemma 14. *Given an instance $(\mathcal{P}, \mathcal{C})$ of the segment cover problem, without a loss of generality we can assume that no segment covers a superset of what another segment covers. That is, for any distinct $A, B \in \mathcal{P}$, we have $A \cap \mathcal{C} \not\subseteq B \cap \mathcal{C}$ and $A \cap \mathcal{C} \not\supseteq B \cap \mathcal{C}$.*

Proof. Trivial. □

Lemma 15. *Given an instance $(\mathcal{P}, \mathcal{C})$ of the segment cover problem, if there exists a line L with at least $k + 1$ points on it, then there exists a subset $\mathcal{A} \subseteq \mathcal{P}$, $|\mathcal{A}| \leq k$, such that every solution \mathcal{R} with $|\mathcal{R}| \leq k$ satisfies $|\mathcal{A} \cap \mathcal{R}| \geq 1$. Moreover, such a subset can be found in polynomial time.*

Proof. First we use Lemma 14.

Let us enumerate the points from \mathcal{C} that lie on L as x_1, x_2, \dots, x_t in the order in which they appear on L . Every segment that is not collinear with L can cover at most one of these points. Therefore, in any solution of size not larger than k , among any k of these points at least one must be covered with segment collinear with L .

Therefore, every solution needs to take one of the segments collinear with L that covers any of the points x_1, x_2, \dots, x_k . After using reduction from Lemma 14, there are at most k such segments that are distinct. □

We are ready to prove Theorem 3.

Proof of Theorem 3.

We will prove this theorem by presenting a branching algorithm that works in desired complexity. It branches over the choice of segments to cover lines with *a lot* of points, then finally solving the small instance, where every line has at most k points by checking all possible solutions.

Algorithm. First we use Lemma 14.

Next, we present a recursive algorithm. Given an instance of the problem:

- (1) If there exist a line with at least $k + 1$ points from \mathcal{C} , we branch over adding to the solution one of the at most k possible segments provided by Lemma 15; name this segment S . Then we find a solution \mathcal{R} for the problem for points $\mathcal{C} - S$, segments $\mathcal{P} - \{S\}$, and parameter $k - 1$. We return $\mathcal{R} \cup \{S\}$.
- (2) If every line has at most k points on it and $|\mathcal{C}| > k^2$, then answer NO.
- (3) If $|\mathcal{C}| \leq k^2$, solve the problem by brute force: check all subsets of \mathcal{P} of size at most k .

475 **Correctness.** Lemma 15 proves that at least one segment that we branch over in (1)
 476 must be present in every solution \mathcal{R} with $|\mathcal{R}| \leq k$. Therefore, the recursive call can find a
 477 solution, provided there exists one.

478 In (2) the answer is no, because every line covers no more than k points from \mathcal{C} , which
 479 implies the same about every segment from \mathcal{P} . Under this assumption we can cover only k^2
 480 points with a solution of size k , which is less than $|\mathcal{C}|$.

481 Checking all possible solutions in (3) is trivially correct.

482 **Complexity.** In the leaves of recursion we have $|\mathcal{C}| \leq k^2$, so $|\mathcal{P}| \leq k^4$, because every
 483 segments can be uniquely identified by the two extreme points it covers (by Lemma 14).
 484 Therefore, there are $\binom{k^4}{k}$ possible solutions to check, each can be checked in time $O(k|\mathcal{C}|)$.
 485 Therefore, (3) takes time $k^{O(k)}$.

486 In this branching algorithm our parameter k is decreased with every recursive call, so we
 487 have at most k levels of recursion with branching over k possibilities. Candidates to branch
 488 over can be found on each level in time $O((nm)^2)$.

489 Reduction from Lemma 14 can be implemented in time $O(n^2m)$.

490 It follows that the overall complexity is $O((nm)^2 \cdot k^{O(k)})$ □

491 4.2. FPT for weighted segments with δ -extensions

492 TODO: Some intro

493 **Theorem 4** (FPT for weighted segment cover with δ -extensions). *There exists an algorithm*
 494 *that given a family \mathcal{P} of n weighted segments (in any direction), a set of m points \mathcal{C} , and*
 495 *parameters k and $\delta > 0$, runs in time $f(k, \delta) \cdot (nm)^c$ for some computable function f and a*
 496 *constant c , and outputs a set $\mathcal{R} \subseteq \mathcal{P}$ such that $|\mathcal{R}| \leq k$ and $\mathcal{R}^{+\delta}$ covers all points in \mathcal{C} , or*
 497 *determines that such a set \mathcal{R} does not exist.*

498 To solve this problem we will introduce a lemma about choosing a *good* subset of points.

499 TODO: Some intuition

500 **Definition 9.** For a set of collinear points C , a subset $A \subseteq C$ is (k, δ) -**good** if for any set of
 501 segments R that covers A and such that $|R| \leq k$, it holds that $R^{+\delta}$ covers C .

502 **Lemma 16.** *There exists an algorithm that for any set of collinear points C , $\delta > 0$ and $k \geq 1$,*
 503 *outputs a (k, δ) -good set $A \subseteq C$ of size at most $f(k, \delta)$ for some computable function f . This*
 504 *algorithm runs in time $O(|C| \cdot f(k, \delta))$.*

505 *Proof.* We prove this for a fixed δ by induction over k .

506 **Inductive hypothesis.** For any set of collinear points C , there exists an algorithm that
 507 runs in time $O(|C|k(1 + \frac{1}{\delta}))$ and finds a set A such that:

- 508 • A is (ℓ, δ) -good for every $1 \leq \ell \leq k$,
- 509 • A has size $|A| < f(\delta, k)$ for some computable function f ,
- 510 • extreme points from C are in A .

Base case for $k = 1$. It is sufficient that A consists of 2 points: extreme points from C or a single point if $|C| = 1$.

If they are covered with one segment, it must be a segment that includes the extreme points from C , so it covers the whole set C .

Inductive step. Assuming inductive hypothesis for any set of collinear points C and for parameter k , we will prove hypothesis for $k + 1$.

Let be s the minimal segment that includes all points from C . That is, the extreme points of C are endpoints of s .

We define $M = \lceil 1 + \frac{2}{\delta} \rceil$ subsegments of s in the following way. We split s into M parts v_i of equal length, that is $|v_i| = \frac{|s|}{M}$ for each $1 \leq i \leq M$.

Let C_i be the subset of C consisting of points laying on v_i .

Let t_i be the segment with endpoints being the extreme points of C_i (it might be degenerated segment if $|C_i| = 1$ or it might be empty if C_i is empty).

TODO: Add a picture with v_i and t_i here

We use the inductive hypothesis to choose (k, δ) -good sets A_i for sets C_i . Note that if $|C_i| \leq 1$, then $A_i = C_i$ and it's still a (k, δ) -good set for C_i .

Then we define $A = \bigcup_{i=1}^M A_i$. Thus A includes the extreme points of C , because they are included in the sets A_1 and A_M .

Proof that A is (k, δ) -good for C . Let us take any cover of A with $k + 1$ segments and call it \mathcal{R} .

For every segment t_i , if there exists a segment x in \mathcal{R} that is disjoint with t_i , then we have a cover of A_i with at most k segments using $\mathcal{R} - \{x\}$. Since A_i is (k, δ) -good for t_i and C_i , then $(\mathcal{R} - \{x\})^{+\delta}$ covers C_i . So $\mathcal{R}^{+\delta}$ covers C_i as well.

If there exists a segment t_i for which a segment x as defined above does not exist, then all $k + 1$ segments that cover A_i intersect with t_i . (Note: There may exist only one such segment t_i). From the inductive hypothesis end points of s are in A_1 and A_M respectively, so \mathcal{R} must cover them. Hence there must exist segments starting in the ends of s and ending somewhere in t_i . Let us call these two segments y and z . It follows that: $|y| + |z| + |t_i| \geq |s|$. Since $|t_i| \leq |v_i| = \frac{|s|}{M} \leq \frac{|s|}{1 + \frac{2}{\delta}} = \frac{|s|\delta}{\delta + 2}$, we have $\max(|y|, |z|) \geq |s|(1 - \frac{\delta}{\delta + 2})/2 = \frac{|s|}{\delta + 2}$.

TODO: Add a picture with such segments here

After δ -extension, the longer of these segments will lengthen both ways by at least:

$$\frac{|s|\delta}{\delta + 2} = \frac{|s|}{1 + \frac{2}{\delta}} > \frac{|s|}{M} = v_i > t_i.$$

Therefore the longer of segments y and z will cover the segment t_i after δ -extension, therefore $\mathcal{R}^{+\delta}$ covers C_i .

Since $C = \bigcup_{i=1}^M C_i$, then $\mathcal{R}^{+\delta}$ covers C .

Complexity We use the recursive algorithm for subsets C_i . Every point from C belongs to at most 2 sets C_i .

Apart from recursive algorithm we perform operations linear in size of $|C| + M$ to calculate the sets C_i .

Therefore it has complexity:

$$O(|C| + M) + \sum_i^M O(|C_i|k(1 + \frac{1}{\delta})) = O(|C| + (1 + \frac{1}{\delta})) + O((\sum_i^M |C_i|)k(1 + \frac{1}{\delta})) \leq O(|C|k(1 + \frac{1}{\delta})).$$

549 *Proof of Theorem 4.* To construct an algorithm for this problem let us formulate some claims
 550 about the problem first.

551 **Definition 10.** Line is **long** if there are at least $k + 1$ points from \mathcal{C} on it.

552 **Claim 1.** *If there are more than k long lines, then \mathcal{C} can not be covered with k segments.*

553 **Claim 2.** *If there is more than k^2 points from \mathcal{C} that do not lie on any long line, then \mathcal{C} can*
 554 *not be covered with k segments.*

555 Applying the above claims, if we have more than k long lines or more than k^2 points form
 556 \mathcal{C} that do not lie on any long line, then we answer that there is no solution of size at most k .

557 Otherwise, we can split \mathcal{C} into at most $k + 1$ sets: D , at most k^2 points that do not lie on
 558 any long line and C_i – points that lay on i -th long line. Sets C_i do not need to be disjoint.

559 Then for every set C_i , we can use Lemma 16 to get (k, δ) -good set A_i for C_i .

560 Then we have set $D \cup \bigcup A_i$ of size at most $f(k, \delta)$ for some computable function f , that
 561 if we have a solution \mathcal{R} of size at most k that covers $D \cup \bigcup A_i$, then $\mathcal{R}^{+\delta}$ covers \mathcal{C} . This is
 562 because \mathcal{R} already covers points D , they cover C_i , because they cover (k, δ) -good set A_i with
 563 at most k segments, so $\mathcal{R}^{+\delta}$ covers C_i .

564 After that we shrunk down size of \mathcal{C} to size of $f(k, \delta)$ for some computable function f .
 565 Then we would like to shrink down size of \mathcal{P} . For every collinear subset of D , we can choose
 566 one segment from \mathcal{P} that covers these points and have the lowest weight or decide there is
 567 no segment that cover them. There are at most $|D|^2$ different segments, because we can
 568 distinguish these collinear sets by their extreme points.

569 This has complexity $O(|D|^2|\mathcal{P}|)$ and produce shrunk down set \mathcal{P} of size $f(k, \delta)$ for some
 570 computable function f .

571 Then we can iterate over all subsets of shrunk down set \mathcal{P} and choose the set with the
 572 lowest sum of weights that cover D . This solution would have weight not larger than optimal
 573 solution for the problem without extension, because we iterate over all possibilities of covering
 574 the subset of \mathcal{C} .

Chapter 5

W[1]-completeness for weighted segments in 3 directions

5.1. W[1]-completeness for weighted segments in 3 directions

Theorem 5. *Weighted geometric set cover with segments in 3 directions is W[1]-hard. Consider the problem of covering a set \mathcal{C} of points by selecting k axis-parallel or right-diagonal weighted segments from a set \mathcal{P} with weights $w : \mathcal{P} \rightarrow \mathbb{R}$ with minimal weight. Assuming ETH, there is no algorithm for this problem with running time $f(k) \cdot (|\mathcal{C}| + |\mathcal{P}|)^{o(\sqrt{k})}$ for any computable function f .*

Corollary 2. *Weighted geometric set cover is W[1]-hard. Assuming ETH, there is no algorithm for weighted geometric set cover with running time $f(k) \cdot (|\mathcal{C}| + |\mathcal{P}|)^{o(\sqrt{k})}$ for any computable function f .*

Proof. Trivial from Theorem 5. □

In order to prove theorem 5 we will show reduction from grid tiling problem.

Definition 11. In the **grid tiling** problem we are given integers n and k , and a function $f : \{1 \dots k\} \times \{1 \dots k\} \rightarrow \mathcal{P}(\{1 \dots n\} \times \{1 \dots n\})$ specifying the set of allowed tiles for each cell of a $k \times k$ grid. The task is to find functions $x, y : \{1 \dots k\} \rightarrow \{1 \dots n\}$ that assign numbers from $\{1 \dots n\}$ to respectively rows and columns of the grid, so that $(x(i), y(j)) \in f(i, j)$ for all valid i and j , or conclude that such assignment does not exist.

Theorem 6. *Assuming ETH, there is no algorithm for grid tiling problem $f(k) \cdot n^{o(\sqrt{k})}$ for any computable function f .*

TODO: proof from reference in literature

Let us have an instance of grid tiling problem – size of the grid k , number of colors n and function of allowed tiles $f : \{1 \dots k\} \times \{1 \dots k\} \rightarrow \{1 \dots n\} \times \{1 \dots n\}$.

Construction. We construct a set \mathcal{P} of segments and a set \mathcal{C} of points.

First let us choose any ordering of n^2 elements and denote it with bijective function $order : \{1 \dots n\} \times \{1 \dots n\} \rightarrow \{1 \dots n^2\}$.

Define $match_v(i, j)$ and $match_h(i, j)$ as functions denoting whether two points share x or y coordinate.

$$match_v(i, j) \iff order^{-1}(i) = \{x_i, y_i\} \wedge order^{-1}(j) = \{x_j, y_j\} \wedge x_i = x_j$$

$$match_h(i, j) \iff order^{-1}(i) = \{x_i, y_i\} \wedge order^{-1}(j) = \{x_j, y_j\} \wedge y_i = y_j$$

Points. Define points:

$$h_{i,j,t} = (i \cdot (n^2 + 1) + t, j \cdot (n^2 + 1))$$

$$v_{i,j,t} = (i \cdot (n^2 + 1), j \cdot (n^2 + 1) + t)$$

Let's define sets H and V as:

$$H = \{h_{i,j,t} : 1 \leq i, j, \leq k, 1 \leq t \leq n^2\}$$

$$V = \{v_{i,j,t} : 1 \leq i, j, \leq k, 1 \leq t \leq n^2\}$$

Let us define $\epsilon = \frac{1}{2k^2}$. For a point $p = \{x, y\}$ we define points:

$$p^L = \{x - \epsilon, y\},$$

$$p^R = \{x + \epsilon, y\},$$

$$p^U = \{x, y + \epsilon\},$$

$$p^D = \{x, y - \epsilon\}.$$

Then we define:

$$\mathcal{C} := H \cup \{p^L : p \in H\} \cup \{p^R : p \in H\} \cup V \cup \{p^U : p \in V\} \cup \{p^D : p \in V\}$$

605

Segments. Define horizontal segments.

$$\text{hor}_{i,j,t_1,t_2} = (h_{i,j,t_1}^R, h_{i+1,j,t_2}^L)$$

$$\text{ver}_{i,j,t_1,t_2} = (v_{i,j,t_1}^D, v_{i,j+1,t_2}^U)$$

$$\text{horBeg}_{i,t} = (h_{1,i,1}^L, h_{1,i,t}^L)$$

$$\text{horEnd}_{i,t} = (h_{k,i,t}^R, h_{k,i,n^2}^R)$$

$$\text{verBeg}_{i,t} = (v_{i,1,1}^U, v_{i,1,t}^U)$$

$$\text{verEnd}_{i,t} = (v_{i,k,t}^D, v_{i,k,n^2}^D)$$

$$\begin{aligned} HOR &= \{\text{hor}_{i,j,t_1,t_2} : 1 \leq i < k, 1 \leq j \leq k, 1 \leq t_1, t_2 \leq n^2, match_h(t_1, t_2)\} \\ &\cup \{\text{horBeg}_{i,t} : 1 \leq i \leq k, 1 \leq t \leq n^2\} \\ &\cup \{\text{horEnd}_{i,t} : 1 \leq i \leq k, 1 \leq t \leq n^2\} \end{aligned}$$

$$\begin{aligned}
VER &= \{\text{ver}_{i,j,t_1,t_2} : 1 \leq i \leq k, 1 \leq j < k, 1 \leq t_1, t_2 \leq n^2, \text{match}_v(t_1, t_2)\} \\
&\cup \{\text{verBeg}_{i,t} : 1 \leq i \leq k, 1 \leq t \leq n^2\} \\
&\cup \{\text{verEnd}_{i,t} : 1 \leq i \leq k, 1 \leq t \leq n^2\}
\end{aligned}$$

$$DIAG := \{(h_{i,j,t}, v_{i,j,t}) : 1 \leq i, j \leq k, 1 \leq t \leq n^2, a_t \in f(i, j)\}$$

606 TODO: explain that these segments are in fact diagonal

$$\mathcal{P} := HOR \cup VER \cup DIAG$$

607 Weight function is equal to length of the segment for HOR and VER and equal to $\delta = \frac{1}{4k^4}$
608 for $DIAG$.

$$w(s) = \begin{cases} |s| & \text{if } s \in HOR \cup VER \\ \delta & \text{if } s \in DIAG \end{cases}$$

609 **Lemma 17.** *If there exists solution for grid tiling, then there exists solution of instance*
610 *$(\mathcal{C}, \mathcal{P}, w)$ of geometric set cover with weight $2k^2(n^2 + 1) - 2k^2\epsilon + k^2\delta - 4k(1 - \epsilon)$.*

Proof. If there exists a solution to the grid tiling problem x, y , then there exists a solution that covers all points

$$\{h_{i,j,t} : 1 \leq i, j \leq k, t = \text{order}(x(i), y(j))\} \cup \{v_{i,j,t} : 1 \leq i, j \leq k, t = \text{order}(x(i), y(j))\}$$

611 with k^2 segments from $DIAG$ and the rest in VER or HOR . This solution has weight
612 $2k^2(n^2 + 1) - 2k^2\epsilon + k^2\delta - 4k(1 - \epsilon)$. \square

613 **Claim 3.** *Points p^L, p^R, p^U, p^D cannot be covered with any segments from $DIAG$.*

614 **Claim 4.** *Points in $H \cup \{p^L : p \in H\} \cup \{p^R : p \in H\}$ have to be covered with segments from*
615 *HOR .*

616 *Points in $V \cup \{p^U : p \in V\} \cup \{p^D : p \in V\}$ have to be covered with segments from VER .*

617 **Claim 5.** *For given i, j two points h_{i,j,t_1}, h_{i,j,t_2} (v_{i,j,t_1}, v_{i,j,t_2}) for $1 \leq t_1 < t_2 \leq n^2$ are covered*
618 *with $DIAG$, then one of them had to be also covered with a segment from HOR (VER).*

619 *Proof.* Assume point h_{i,j,t_1} is not covered with segments from HOR . Point h_{i,j,t_1}^R has to be
620 covered with HOR from Claim 4. And every segment in HOR covering h_{i,j,t_1}^R , covers also
621 h_{i,j,t_2} . \square

622 **Lemma 18.** *Any solution has to have sum of weights from HOR and VER at least $2k^2(n^2 +$
623 $1) - 2k^2\epsilon - 4k(1 - \epsilon)$. Denote this weight as W_{hv} .*

624 *Proof.* • We know that for every i, j only one $v_{i,j,t}$ and $h_{i,j,t}$ can be uncovered by HOR
625 and VER (Claim 5),

626 • We sum the lower bound for sum of length for horizontal/vertical lines for a single
627 vertical line and:

628 1. it is k sections, each of length $(n^2 + 1)$, one $v_{i,j,t}$ is not covered for each of k segments,

629 2. its guards $v_{i,j,t}^U$ and $v_{i,j,t}^D$ have to be covered in VER (Claim 3), so it leaves 2ϵ free
630 space

3. Length between $v_{i,1,1}^D$ and v_{i,k,n^2}^U is

$$(k-2)(n^2+1) + 2(n^2+\epsilon) = k(n^2+1) - 2(1-\epsilon).$$

631 • therefore sum of these segments are $2k^2(n^2+1) - 2k^2\epsilon - 4k(1-\epsilon)$

632 \square

633 **Lemma 19.** *Solution has to have at most one DIAG segment in every square and it is a*
634 *correct tile in grid tiling.*

635 *Proof.* • At most one h and v points are covered with DIAG for i,j (Claim 5)

- Exactly one h and v points are covered with DIAG for every i, j , because if one of them were not, then we would use too much weight

$$2k^2(n^2+1) - 2k^2\epsilon - 4k(1-\epsilon) + 2\epsilon > 2k^2(n^2+1) - 2k^2\epsilon + k^2\delta - 4k(1-\epsilon)$$

636 • This points are covered with the same segment from DIAG, because we need to use at
637 least k^2 of them to use exactly one DIAG segment for every i, j , if we used 2 DIAGs for
638 one of (i, j) , then we would have used too much weight by δ

639 • Therefore t_x and t_y are allowed tile for i,j, because respective segment was in DIAG

640 \square

641 **Lemma 20.** *Points from previous point are synchronized in one row/column $match_h(t_1, t_2)$*
642 *and $match_v(t_1, t_2)$ must be true*

643 *Proof.* • Every space between points in H and V covered by HOR/VER has to be covered
644 by only one segment, because otherwise it would use $\epsilon > k^2\delta$ additional weight, which
645 is larger than weight of solution

646 • Therefore for every t_1 and t_2 that are consecutive uncovered points, there must be a
647 segment connecting h_{i,j,t_1}^R and h_{i+1,j,t_2}^L , because these points have to be covered in HOR
648 (Claim 3), therefore $match_h(t_1, t_2)$ must be true

649 \square

650 **Lemma 21.** *If there exists solution of instance $(\mathcal{C}, \mathcal{P}, w)$ of geometric set cover with weight*
651 *at most $2k^2(n^2+1) - 2k^2\epsilon + k^2\delta - 4k(1-\epsilon)$, then there exists a solution for grid tiling.*

652 *Proof.* Therefore we take the points that are not covered in the solution that we have and we
653 know that for each row and column $order^{-1}(v_{i,*}) = x_i$ and we set $x(i) = x_i$, $order^{-1}(h_{*,i}) = y_i$
654 and we set $y(i) = y_i$. This is a solution to grid tiling problem. \square

655 *Proof of Theorem 5.* Based on Lemmas 17 and 21 this is true. \square

656 5.2. What is missing

657 We don't know FPT for axis-parallel segments without δ -extensions.

Chapter 6

Geometric Set Cover with lines

6.1. Lines parallel to one of the axis

When \mathcal{R} consists only of lines parallel to one of the axis, the problem can be solved in polynomial time.

We create bipartial graph G with node for every line on the input split into sets: H – horizontal lines and V – vertical lines. If any two lines cover the same point from \mathcal{C} , then we add edge between them.

Of course there will be no edges between nodes inside H , because all of them are pararell and if they share one point, they are the same lines. Similar argument for V . So the graph is bipartial.

Now Geometric Set Cover can be solved with Vertex Cover on graph G . Since Vertex Cover (even in weighted setting) on bipartial graphs can be solved in polynomial time.

Short note for myself just to remember how to this in polynomial time:

Non-weighted setting - Konig theorem + max matching

Weighted setting - Min cut in graph of $\neg A$ or $\neg B$ (edges directed from V to H)

6.2. FPT for arbitrary lines

You can find this is Platypus book. We will show FPT kernel of size at most k^2 .

(Maybe we need to reduce lines with one point/points with one line).

For every line if there is more than k points on it, you have to take it. At the end, if there is more than k^2 points, return NO. Otherwise there is no more than k^4 lines.

In weighted settings among the same lines with different weights you leave the cheapest one and use the same algorithm.

6.3. APX-completeness for arbitrary lines

We will show a reduction from Vertex Cover problem. Let's take an instance of the Vertex Cover problem for graph G . We will create a set of $|V(G)|$ pairwise non-pararell lines, such that no three of them share a common point.

Then for every edge in $(v, w) \in E(G)$ we put a point on crossing of lines for vertices v and w . They are not pararell, so there exists exactly one such point and any other line don't cover this point (any three of them don't cross in the same point).

Solution of Geometric Set Cover for this instance would yield a sound solution of Vertex Cover for graph G . For every point (edge) we need to choose at least one of lines (vertices) v or w to cover this point.

Vertex Cover for arbitrary graph is APX-complete, so this problem is also APX-complete.

6.4. 2-approximation for arbitrary lines

Vertex Cover has an easy 2-approximation algorithm, but here very many lines can cross through the same point, so we can do d -approximation, where d is the biggest number of lines crossing through the same point. So for set where any 3 lines don't cross in the same point it yields 2-approximation.

The problematic cases are where through all points cross at least k points and all lines have at least k points on them. It can be created by casting k -grid in k -D space on 2D space.

Greedy algorithm yields $\log |\mathcal{R}|$ -approximation, but I have example for this for bipartial graph and reduction with taking all lines crossing through some point (if there are no more than k) would solve this case. So maybe it works.

Unfortunately I haven't done this :(

I can link some papers telling it's hard to do.

6.5. Connection with general set cover

Problem with finite set of lines with more dimensions is equivalent to problem in 2D, because we can project lines on the plane which is not perpendicular to any plane created by pairs of (point from \mathcal{C} , line from \mathcal{P}).

Of course every two lines have at most one common point, so is every family of sets that have at most one point in common equivalent to some geometric set cover with lines?

No, because of Desargues's theorem. Have to write down exactly what configuration is banned.

712 Chapter 7

713 Geometric Set Cover with polygons

714 7.1. State of the art

715 Covering points with weighted discs admits PTAS [Li and Jin, 2015] and with fat polygons
716 with δ -extensions with unit weights admits EPTAS [Har-Peled and Lee, 2009].

717 Although with thin objects, even if we allow δ -expansion, the Set Cover with rectangles is
718 APX-complete (for $\delta = 1/2$), it follows from APX-completeness for segments with δ -expansion
719 in Section 3.1.

720 Covering points with squares is W[1]-hard [Marx, 2005]. It can be proven that assuming
721 *SETH*, there is no $f(k) \cdot (|\mathcal{C}| + |\mathcal{P}|)^{k-\epsilon}$ time algorithm for any computable function f and
722 $\epsilon > 0$ that decides if there are k polygons in \mathcal{P} that together cover \mathcal{C} , *Theorem 1.9* in [Marx
723 and Pilipczuk, 2015].

⁷²⁴ Chapter 8

⁷²⁵ Conclusions

726 Bibliography

- 727 [Har-Peled and Lee, 2009] Har-Peled, S. and Lee, M. (2009). Weighted geometric set cover
728 problems revisited. *Journal of Computational Geometry*, 3.
- 729 [Håstad, 2001] Håstad, J. (2001). Some optimal inapproximability results. *J. ACM*,
730 48(4):798–859.
- 731 [Li and Jin, 2015] Li, J. and Jin, Y. (2015). A PTAS for the weighted unit disk cover problem.
732 *CoRR*, abs/1502.04918.
- 733 [Marx, 2005] Marx, D. (2005). Efficient approximation schemes for geometric problems? In
734 Brodal, G. S. and Leonardi, S., editors, *Algorithms – ESA 2005*, pages 448–459, Berlin,
735 Heidelberg. Springer Berlin Heidelberg.
- 736 [Marx and Pilipczuk, 2015] Marx, D. and Pilipczuk, M. (2015). Optimal parameterized algo-
737 rithms for planar facility location problems using voronoi diagrams. *CoRR*, abs/1504.05476.