

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. Geometric Set Cover with segments	9
45	3.1. FPT for segments	9
46	3.1.1. Axis-parallel segments	9
47	3.1.2. Segments in arbitrary directions	9
48	3.2. APX-completeness for segments parallel to axes	11
49	3.2.1. MAX-(3,3)-SAT and statement of reduction	11
50	3.2.2. Reduction	13
51	3.2.2.1. VARIABLE-gadget	13
52	3.2.2.2. OR-gadget	14
53	3.2.2.3. CLAUSE-gadget	16
54	3.2.2.4. Summary	18
55	3.2.2.5. Summary of construction	18
56	3.2.3. Construction lemmas and proof of Lemma 3	19
57	3.3. FPT for weighted segments with δ -extensions	21
58	3.4. W[1]-completeness for weighted segments in 3 directions	24
59	3.5. What is missing	26
60	4. Geometric Set Cover with lines	27
61	4.1. Lines parallel to one of the axis	27
62	4.2. FPT for arbitrary lines	27
63	4.3. APX-completeness for arbitrary lines	27
64	4.4. 2-approximation for arbitrary lines	28
65	4.5. Connection with general set cover	28
66	5. Geometric Set Cover with polygons	29
67	5.1. State of the art	29
68	6. Conclusions	31

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

Geometric Set Cover with segments

3.1. 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.

3.1.1. 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 3.1.2

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

3.1.2. 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 1. (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 1. *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 2. *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 1.

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 1, there are at most k such segments that are distinct. □

We are ready to prove Theorem 1.

Proof of Theorem 1.

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 1.

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 2; 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 .

Correctness. Lemma 2 proves that at least one segment that we branch over in (1) must be present in every solution \mathcal{R} with $|\mathcal{R}| \leq k$. Therefore, the recursive call can find a solution, provided there exists one.

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

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

219 **Complexity.** In the leaves of recursion we have $|\mathcal{C}| \leq k^2$, so $|\mathcal{P}| \leq k^4$, because every
220 segments can be uniquely identified by the two extreme points it covers (by Lemma 1).
221 Therefore, there are $\binom{k^4}{k}$ possible solutions to check, each can be checked in time $O(k|\mathcal{C}|)$.
222 Therefore, (3) takes time $k^{O(k)}$.

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

226 Reduction from Lemma 1 can be implemented in time $O(n^2m)$.

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

228 3.2. APX-completeness for segments parallel to axes

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

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

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

237 Theorem 2 implies the following.

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

240 We prove Theorem 2 by taking a problem that is APX-hard and showing a reduction. For
241 this problem we choose MAX-(3,3)-SAT which we define below.

242 3.2.1. MAX-(3,3)-SAT and statement of reduction

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

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

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

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

252 **Theorem 3.** [Håstad, 2001]

253 *For any $\epsilon > 0$, it is NP-hard to distinguish satisfiable (3,3)-SAT formulas from at most*
254 *$(7/8 + \epsilon)$ -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 3) 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 2.

Lemma 3. *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 3 in subsequent sections, but meanwhile let us prove Theorem 2 using Lemma 3 and Theorem 3.

TODO: This below can't use current template

Proof of Theorem 2.

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 3, 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 3. 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 3 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:

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

From Lemma 3 we have:

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

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

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

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

292 3.2.2. Reduction

293 We proceed to the proof of Lemma 3. That is, we show a reduction from MAX-(3,3)-SAT
294 problem to geometric set cover with segments parallel to axis. Moreover, the obtained instance
295 of geometric set cover will be robust to 1/2-extensions (have the same optimal solution after
296 1/2-extension).

297 The construction will be composed of 2 types of gadgets: **VARIABLE-gadgets** and
298 **CLAUSE-gadgets**. **CLAUSE-gadgets** would be constructed using two **OR-gadgets** con-
299 nected together. Every gadget consists of a point set and a segment set.

300 3.2.2.1. VARIABLE-gadget

301 VARIABLE-gadget is responsible for choosing the value of a variable in a CNF formula. It
302 allows two minimum solutions of size 3 each. These two choices correspond to the two Boolean
303 values of the variable.

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

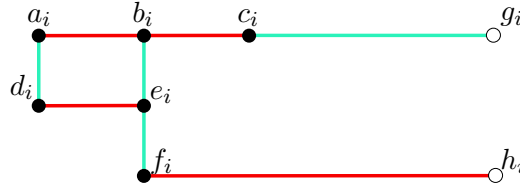


Figure 3.1: **VARIABLE-gadget**. We denote the set of points marked with black circles as pointsVar^i , and they need to be covered (are part of the set \mathcal{C}). Note that some of the points are not marked as black dots and exists only to name segments for further reference. We denote the set of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

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

Let us define:

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

and

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

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

307 **Segments.** Let us define:

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

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

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

308 **Lemma 4.** For any $1 \leq i \leq n$, points in pointsVariable_i can be covered using 3 segments from
309 $\text{segmentsVariable}_i$.

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

311 **Lemma 5.** For any $1 \leq i \leq n$, points in pointsVariable_i can not be covered with fewer than 3
312 segments from $\text{segmentsVariable}_i$.

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

315 **Lemma 6.** For every set $A \subseteq \text{segmentsVariable}_i$ such that A covers pointsVariable_i and
316 $(c_i, g_i), (f_i, h_i) \in A$, it holds that $|A| \geq 4$.

317 *Proof.* No segment from $\text{segmentsVariable}_i$ covers more than one point from $\{a_i, e_i\}$, therefore
318 $\text{pointsVariable}_i - \{c_i, f_i, g_i, h_i\}$ can not be covered with fewer than 2 segments. □

319 3.2.2.2. OR-gadget

320 OR-gadget has 3 important segments – x, y, result . x and y don't count to the weight of
321 solution of OR-gadget (they are part of different gadgets). It has a minimal solution of weight
322 w and result can be chosen only if x or y are also chosen for the solution. If none of them
323 are chosen, then solution choosing result segment has weight at least $w + 1$. Therefore the
324 following formula holds for a solution R assuming that R uses only w from this OR-gadget:

$$(x \in R) \vee (y \in R) \iff \text{result} \in R$$

325 **Points.**

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

$$\text{vec}_{i,j} = (10i + 3 + 3j, 4n + 2j)$$

327 Define $\{l_{i,j}, m_{i,j} \dots v_{i,j}\}$ as $\{l_0, m_0 \dots v_0\}$ shifted by $\text{vec}_{i,j}$

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

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



Figure 3.2: **OR-gadget**. Figure presenting $\text{orGadget}_{i,j}$: segments from $or_{i,j}^{false}$ are red, segments from $or_{i,j}^{true}$ are blue, segments from $\text{orMoveVariable}_{i,j}$ are yellow and green.

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

$$\begin{aligned}
 or_{i,j}^{false} &= \{(q_{i,j}, r_{i,j}), (s_{i,j}, u_{i,j})\} \\
 or_{i,j}^{true} &= \{(m_{i,j}, s_{i,j}), (o_{i,j}, u_{i,j}), (t_{i,j}, v_{i,j})\} \\
 \text{orMoveVariable}_{i,j} &= \{(l_{i,j}, n_{i,j}), (n_{i,j}, p_{i,j})\}
 \end{aligned}$$

330 Segments in OR-gadget:

$$\text{segmentsOrGadget}_{i,j} = or_{i,j}^{false} \cup or_{i,j}^{true} \cup \text{orMoveVariable}_{i,j}$$

331 **Lemma 7.** For any $1 \leq i \leq n, j \in \{0, 1\}$ and $x \in \{l_{i,j}, p_{i,j}\}$ points in $\text{pointsOrGadget}_{i,j} - \{x\} \cup \{v_{i,j}\}$ can be covered with 4 segments from $\text{segmentsOrGadget}_{i,j}$.

333 *Proof.* We can do that using one segment from $\text{orMoveVariable}_{i,j}$ (chosen depending on the value of x) and all segments from $or_{i,j}^{true}$. □

335 **Lemma 8.** For any $1 \leq i \leq n, j \in \{0, 1\}$, points in $\text{pointsOrGadget}_{i,j}$ can be covered with 4 segments from $\text{segmentsOrGadget}_{i,j}$.

337 *Proof.* We can do that using $\text{orMoveVariable}_{i,j}$ and $or_{i,j}^{false}$. □

338 **Lemma 9.** For any $1 \leq i \leq n, j \in \{0, 1\}$ and $x \in \{l_{i,j}, p_{i,j}\}$ points in $\text{pointsOrGadget}_{i,j} - \{x\} \cup \{v_{i,j}\}$ can not be covered with less than 4 segments from $\text{segmentsOrGadget}_{i,j}$.

340 *Proof.* TODO □

341 **Lemma 10.** For any $1 \leq i \leq n, j \in \{0, 1\}$, points in $\text{pointsOrGadget}_{i,j}$ can not be covered
 342 with less than 4 segments from $\text{segmentsOrGadget}_{i,j}$.

343 *Proof.* TODO □

344 3.2.2.3. CLAUSE-gadget

345 CLAUSE-gadget is responsible for calculating if choice of the variable values meets the clause
 346 in formula. It has minimal solution of weight w if at least one variable in the clause has a
 347 correct value. Otherwise it has minimal solution $w + 1$. This way by the minimal solution for
 348 the whole problem, we can tell how many clauses were satisfiable.

349 The CLAUSE-gadgets consist of two OR-gadgets. We don't want the CLAUSE-gadgets
 350 to be crammed somewhere between the very long variable segments. That's why we have a
 351 simple gadget to *pass* the value of the segment, ie. segments $(x_{i,0}, x_{i,1}), (y_{i,0}, y_{i,1}), (z_{i,0}, z_{i,1})$.
 352 Two segments and one of them is chosen if x was chosen in the solution and the other one if
 353 x wasn't.

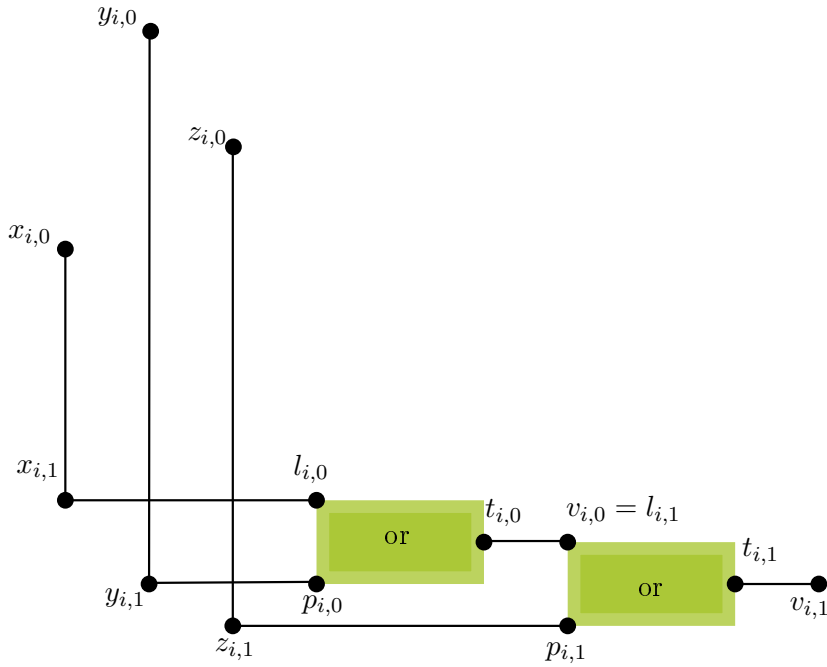


Figure 3.3: **CLAUSE-gadget.** We denote set of these points as pointsClause_i . Every green rectangle is an OR-gadget. y -coordinates of $x_{i,0}$, $y_{i,0}$ and $z_{i,0}$ depend on the values of variables in the i -th clause.

354 **Points.** TODO: Rephrase it

355 Assuming clause $C_i = x_i \vee y_i \vee z_i$, function $\text{idx}(w)$ is returning index of the variable w ,
 356 function $\text{neg}(w)$ is returning whether variable w is negated in a clause.

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

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

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

Segments.

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

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

360 *Proof.* For $a = x_{i,0}$ (analogous proof for $y_{i,0}$): First we use Lemma 7 twice with excluded
361 $x = l_{i,0}$ and $x = l_{i,1} = v_{i,0}$, resulting with 8 segments $\text{or}_{i,0}^{\text{true}} \cup \text{or}_{i,1}^{\text{true}}$ which cover all required
362 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:
363 $\{(x_{i,1}, l_{i,0}), (y_{i,0}, y_{i,1}), (z_{i,0}, z_{i,1})\}$

364 For $a = z_{i,0}$: Using Lemma 8 and Lemma 7 with $x = p_{i,1}$, resulting with 8 segments
365 $\text{or}_{i,0}^{\text{false}} \cup \text{or}_{i,1}^{\text{true}}$ 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
366 those using additional 3 segments: $\{(x_{i,0}, x_{i,1}), (y_{i,0}, y_{i,1}), (z_{i,1}, p_{i,1})\}$. \square

367 **Lemma 12.** *For any $1 \leq i \leq n$ there is $\text{solClause}_i^{\text{false}} \subset \text{segmentsClause}_i$ with $|\text{solClause}_i^{\text{false}}| =$
368 12 that covers points in pointsClause_i .*

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

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

372 **Lemma 13.** *For any $1 \leq i \leq n$:*

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

375 (2) *points in pointsClause_i can not be covered using any subset of segments from segmentsClause_i*
376 *of size smaller than 12.*

377 *Proof of no cover with fewer than 12 segments.* There is independent set of 12 points in $\text{pointsClause}_i \supseteq$
378 $\{x_{i,0}, y_{i,0}, z_{i,0}, l_{i,0}, p_{i,0}, q_{i,0}, u_{i,0}, v_{i,0} = l_{i,1}, p_{i,1}, q_{i,1}, u_{i,1}, v_{i,1}\}$. \square

379 *Proof of no cover with fewer than 11 segments.* We can choose disjoint sets X, Y, Z such that
380 $X \cup Y \cup Z \subseteq \text{pointsClause}_i - \{x_{i,0}, y_{i,0}, z_{i,0}\}$ and there are no segments covering points from
381 different sets. And we prove lower bounds for each of these sets.

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

382 Set X is an indendent set, so it must be covered with 3 segments.

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

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

383 For both Y and Z we can check all of the subsets of 3 segments with brutforce that none
384 of them cover, so they have to be covered with 4 segments.

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

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

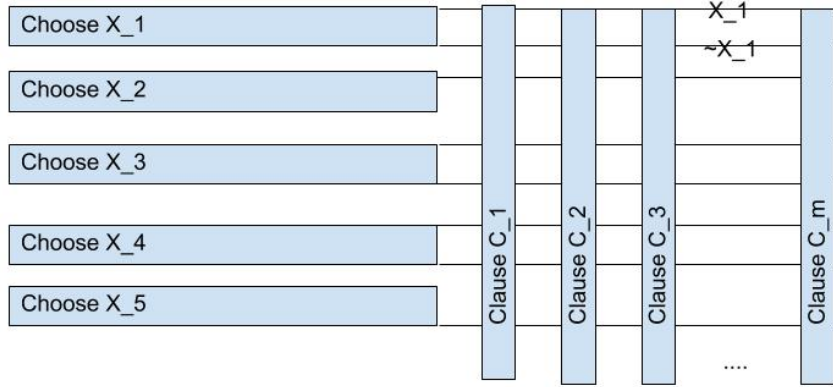


Figure 3.4: **General schema.**

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.

3.2.2.4. Summary

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

Lemma 14. 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

□

3.2.2.5. 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.

396 We prove some properties of different gadgets. Every segment for a gadget will only cover
 397 points in this gadget (won't interact with any different gadget), so we can prove lemmas *locally*.
 398 TODO: y axis is increasing values downward on figures (not upwards like in normal).

399 3.2.3. Construction lemmas and proof of Lemma 3

400 In order to prove Lemma 3 we introduce some auxiliary lemmas proving properties of the
 401 construction described in the previous section.

402 Consider an instance S of MAX-(3,3)-SAT of size n with optimum solution satisfying k
 403 clauses. Let us construct an instance $(\mathcal{C}, \mathcal{P})$ of geometric set cover as described in Section 3.2.2
 404 for instance S of MAX-(3,3)-SAT.

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

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

408 We cover every VARIABLE-gadget with solution described in Lemma 4, in the i -th gadget
 409 choosing the set of segments corresponding to the value of $\phi(x_i)$.

410 For every clause that is satisfied, say c_i , let us name the variable that is **true** in it x_i and
 411 point corresponding to x_i in **pointsClause_i** as a . Points in **pointsClause_i** are covered with set
 412 **solClause_i^{true,a}** described in Lemma 11. For every clause that is not satisfied, say c_j , points in
 413 **pointsClause_j** are covered with set **solClause_j^{false}** described in Lemma 12.

414 TODO: Some text about what we define here

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

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

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

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

418 □

419 **Lemma 16.** *Suppose we have a solution \mathcal{R} of the instance $(\mathcal{C}, \mathcal{P})$ of geometric set cover that
 420 is of size w . Then there exists a solution of S that satisfies at least $15n - w$ clauses.*

421 *Proof.*

422 Let the clauses in S be $c_1, c_2 \dots c_n$ and variables be $x_1, x_2 \dots x_n$. Given a solution \mathcal{R}
 423 of the instance $(\mathcal{C}, \mathcal{P})$ of geometric set cover, we construct a solution of S by constructing
 424 an assignment of variables $\phi : \{x_1, x_2 \dots x_n\} \rightarrow \{\mathbf{true}, \mathbf{false}\}$ that satisfies at least $15n - w$
 425 clauses in S .

426 **Variables** The solution \mathcal{R} needs to use at least 3 segments to cover pointsVariable_i for
 427 each VARIABLE-gadget (Lemma 5). If \mathcal{R} contains both segments (c_i, g_i) and (f_i, h_i) , then it
 428 has used at least 4 segments (Lemma 6).

$$\begin{cases} |\text{pointsVar}_i \cap \mathcal{R}| \geq 4 & \text{if } (c_i, g_i) \in \mathcal{R} \wedge (f_i, h_i) \in \mathcal{R} \\ |\text{pointsVar}_i \cap \mathcal{R}| \geq 3 & \text{otherwise} \end{cases}$$

429 We also create modified solution \mathcal{R}' such that $|\mathcal{R}'| = |\mathcal{R}|$ and \mathcal{R}' is still a correct solution
 430 for $(\mathcal{C}, \mathcal{P})$.

431 If \mathcal{R} contains at most one of the segments (c_i, g_i) and (f_i, h_i) , choose the corresponding
 432 variable value to the constructed assignment ϕ . If \mathcal{R} contains none of these segments, set value
 433 to **false**. If \mathcal{R} contains both segments, assign the value $\phi(x_i)$ that satisfies most clauses in
 434 which x_i occurs. Every variable is in exactly 3 clauses, so one assignment satisfies at least 2 of
 435 them. In this case we remove one of the segments (c_i, g_i) or (f_i, h_i) (depending on the value of
 436 $\phi(x_i)$) from \mathcal{R}' and replace it with a segment that covers the point $v_{j,1}$ for a CLAUSE-gadget
 437 for a clause c_j that uses x_i with value $\phi(x_i)$

Formally, we define the value $\phi(x_i)$ for the variable x_i as follows:

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

438 and \mathcal{R}' as:

$$\mathcal{R}' = \mathcal{R} \begin{cases} -\{(f_i, h_i)\} \cup \{(t_{j,1}, v_{j,1})\} & \phi(x_i) = \text{true} \wedge \text{if } (c_i, g_i) \in \mathcal{R} \wedge (f_i, h_i) \wedge \\ & \text{clause } c_j \text{ is satisfied by } \phi(x_i) = \text{false} \\ -\{(c_i, g_i)\} \cup \{(t_{j,1}, v_{j,1})\} & \phi(x_i) = \text{false} \wedge \text{if } (c_i, g_i) \in \mathcal{R} \wedge (f_i, h_i) \wedge \\ & \text{clause } c_j \text{ is satisfied by } \phi(x_i) = \text{true}, \end{cases}$$

439 therefore $|\text{pointsVar}_i \cap \mathcal{R}'| \geq 3$.

440 TODO: Maybe remove section below, because we do this calculation at the end anyway
 441 To cover $\bigcup_{1 \leq i \leq n} \text{pointsVar}_i$ we have used at least $3n + a$ segments, where a is the number of
 442 i such that \mathcal{R} consists of values (c_i, g_i) and (f_i, h_i) .

443 **Clauses** For a clause $C_i = x \vee y \vee z$, \mathcal{R}' needs to use at least 11 segments to cover
 444 $\text{pointsClause}_i - \{x, y, z\}$ in CLAUSE-gadget (Lemma 13).

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

446 Moreover, if all of the points $\{x_{i,0}, y_{i,0}, z_{i,0}\}$ are not covered by the segments from $\mathcal{R}' \cap$
 447 pointsVariable_i , then \mathcal{R}' needs to cover pointsClause_i with at least 12 segments by Lemma 13.

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:

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

448 where b is the number of clauses where none of the segments covering the points $x_{i,0}, y_{i,0}, z_{i,0}$
 449 were chosen in P_{var}^j .

450 **Satisfied clauses with chosen variable assignment.** Consider a clause, say c_i .
 451 If none of the points $x_{i,0}, y_{i,0}, z_{i,0}$ in pointsClause_i were covered by segments from $\mathcal{R}' \cap$
 452 $\text{segmentsVariable}_j$, this clause is not satisfied by assignment ϕ .

453 If one of these points is covered by segments from VARIBALE-gadget (TODO better this
 454 or $\mathcal{R}' \cap \text{segmentsVariable}_j$), then denote this point as t and say it corresponds to variable x_j .
 455 Consider the cases of choosing value of $\phi(x_j)$ in equation (3.1).

456 If \mathcal{R}' contains only one of the segments (c_j, g_j) and (f_j, h_j) , then the value $\phi(x_j)$ satisfies
 457 c_i .

458 If \mathcal{R}' contains neither (c_j, g_j) nor (f_j, h_j) , then it is impossible that t is covered by segments
 459 in $\mathcal{R} \cap \text{segmentsVariable}_j$.

460 \mathcal{R}' was chosen in a way, that it is impossible that it contains both (c_i, g_i) and (f_i, h_i) for
 461 some variable i .

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

463 To conclude, we proved that given a solution to $(\mathcal{C}, \mathcal{P})$ of size w , we have constructed a
 464 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

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

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

466 We are ready to conclude the proof of Lemma 3.

Proof of Lemma 3. By Lemma 15, we know that there exists a solution to $(\mathcal{C}, \mathcal{P})$ of size
 $15n - k$, so:

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

Since the optimum solution of S satisfies k clauses, then according to Lemma 16:

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

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

469 3.3. FPT for weighted segments with δ -extensions

470 TODO: Some intro

471 **Theorem 4** (FPT for weighted segment cover with δ -extensions). *There exists an algorithm*
 472 *that given a family \mathcal{P} of n weighted segments (in any direction), a set of m points \mathcal{C} , and*
 473 *parameters k and $\delta > 0$, runs in time $f(k, \delta) \cdot (nm)^c$ for some computable function f and a*
 474 *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*
 475 *determines that such a set \mathcal{R} does not exist.*

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

477 TODO: Some intuition

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

480 **Lemma 17.** *There exists an algorithm that for any set of collinear points C , $\delta > 0$ and $k \geq 1$,*
 481 *outputs a (k, δ) -good set $A \subseteq C$ of size at most $f(k, \delta)$ for some computable function f . This*
 482 *algorithm runs in time $O(|C| \cdot f(k, \delta))$.*

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

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

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

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

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

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

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

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

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

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

502 TODO: Add a picture with v_i and t_i here

503 We use the inductive hypothesis to choose (k, δ) -good sets A_i for sets C_i . Note that if
 504 $|C_i| \leq 1$, then $A_i = C_i$ and it's still a (k, δ) -good set for C_i .

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

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

509 For every segment t_i , if there exists a segment x in \mathcal{R} that is disjoint with t_i , then we have
 510 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 ,
 511 then $(\mathcal{R} - \{x\})^{+\delta}$ covers C_i . So $\mathcal{R}^{+\delta}$ covers C_i as well.

512 If there exists a segment t_i for which a segment x as defined above does not exist, then all
 513 $k + 1$ segments that cover A_i intersect with t_i . (Note: There may exist only one such segment
 514 t_i). From the inductive hypothesis end points of s are in A_1 and A_M respectively, so \mathcal{R} must
 515 cover them. Hence there must exist segments starting in the ends of s and ending somewhere
 516 in t_i . Let us call these two segments y and z . It follows that: $|y| + |z| + |t_i| \geq |s|$. Since
 517 $|t_i| \leq |v_i| = \frac{|s|}{M} \leq \frac{|s|}{1 + \frac{2}{\delta}} = \frac{|s|\delta}{\delta + 2}$, we have $\max(|y|, |z|) \geq |s|(1 - \frac{\delta}{\delta + 2})/2 = \frac{|s|}{\delta + 2}$.

518 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:

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

□

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

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

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

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 not be covered with k segments.*

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

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

Then for every set C_i , we can use Lemma 17 to get (k, δ) -good set A_i for C_i .

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

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

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

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

□

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

Theorem 5. *W[1]-completeness for weighted segments in 3 directions.* Consider the problem of covering a set \mathcal{C} of points by selecting k axis-parallel or right-diagonal weighted segments with weights from a set \mathcal{P} 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 .

We will show reduction from grid tiling problem.

Let's have an instance of grid tiling problem – size of the grid k , number of elements available n and k^2 sets of available pairs in every tile $S_{i,j} \subseteq \{1, n\} \times \{1, n\}$.

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

First let's choose any ordering of n^2 elements $\{1, n\} \times \{1, n\}$ and name this sequence $a_1 \dots a_{n^2}$.

$$\text{match}_v(i, j) \iff a_i = \{x_i, y_i\} \wedge a_j = \{x_j, y_j\} \wedge x_i = x_j$$

$$\text{match}_h(i, j) \iff a_i = \{x_i, y_i\} \wedge a_j = \{x_j, y_j\} \wedge y_i = y_j$$

Points. Define points:

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

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

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's define $\epsilon = 0.1$. For a point $\{x, y\} = p$ we define points $p^L = \{x - \epsilon, y\}$, $p^R = \{x + \epsilon, y\}$, $p^U = \{x, y - \epsilon\}$, and $p^D = \{x, y + \epsilon\}$.

Then we define:

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

Segments. Define horizontal segments.

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

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

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

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

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

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

$$\begin{aligned}
HOR &= \{hor_{i,j,t_1,t_2} : 1 \leq i \leq k, 1 \leq j < k, 1 \leq t_1, t_2 \leq n^2, match_h(t_1, t_2)\} \\
&\cup \{horbeg_{i,t} : 1 \leq i \leq k, 1 \leq t \leq n^2\} \\
&\cup \{horend_{i,t} : 1 \leq i \leq k, 1 \leq t \leq n^2\}
\end{aligned}$$

$$\begin{aligned}
VER &= \{ver_{i,j,t_1,t_2} : 1 \leq i \leq k, 1 \leq j < k, 1 \leq t_1, t_2 \leq n^2, match_v(t_1, t_2)\} \\
&\cup \{verbeg_{i,t} : 1 \leq i \leq k, 1 \leq t \leq n^2\} \\
&\cup \{verend_{i,t} : 1 \leq i \leq k, 1 \leq t \leq n^2\}
\end{aligned}$$

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

568 TODO: explain that these segments are in fact diagonal

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

569 **Lemma 18.** *If there exists solution for grid tiling, then there exists solution for our construc-*
570 *tion using $2(k+1)k + k^2$ segments with weight exactly $2k \cdot (k(n^2 + 1) - 2 - 2\epsilon(k-1))$.*

Claim 3. *If there exists a solution to the grid tiling $c_1 \dots c_k$ and $r_1 \dots r_k$, then there exists a solution covering all points*

$$\{h_{i,j,t} : 1 \leq i, j \leq k, t = (c_i, r_j)\} \cup \{v_{i,j,t} : 1 \leq i, j \leq k, t = (c_j, r_i)\}$$

571 *with segments in DIAG and the rest in VER or HOR and has weight $2k \cdot (k(n^2 + 1) -$*
572 *$2 - 2\epsilon(k-1))$.*

573 **Proof.** TODO: jakiś prosty z definicji

574 **Lemma 19.** *If there exists solution for our construction using $2(k+1)k + k^2$ segments with*
575 *weight exactly $2k \cdot (k(n^2 + 1) - 2 - 2\epsilon(k-1))$, then there exists a solution for grid tiling*

576 **Proof.** This follows from Lemma 20, because we just take which points are covered with
577 *DIAG.*

578 **Claim 4.** *Points p^L, p^R, p^U, p^D cannot be covered with DIAG.*

579 **Claim 5.** *Points in $H \cup \{p^L : p \in H\} \cup \{p^R : p \in H\}$ cannot be covered with VER.*

580 *Points in $V \cup \{p^U : p \in V\} \cup \{p^D : p \in V\}$ cannot be covered with HOR.*

581 **Claim 6.** *For given i, j if none of the points $h_{i,j,t}$ ($v_{i,j,t}$) for $1 \leq t \leq n^2$ are covered with*
582 *DIAG, then some spaces between neighbouring points were covered twice.*

583 **Claim 7.** *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*
584 *with DIAG, then one of them had to be also covered with a segment from HOR (VER).*

585 **Proof.** Point v_{i,j,t_2}^L had to be covered with VER from Claims 4 and 5. And every segment
586 *in VER covering v_{i,j,t_2}^L , covers also v_{i,j,t_1}^L .*

587 **Lemma 20.** *If there exists solution for our construction with weight at most (exactly) $2k \cdot$*
588 *$(k(n^2 + 1) - 2 - 2\epsilon(k-1))$, then for every i, j there must be exactly one t such that $h_{i,j,t}$ ($v_{i,j,t}$)*
589 *is covered with DIAG and moreover if h_{i,j,t_1} and $h_{i,j+1,t_2}$ are uncovered, then $match_h(t_1, t_2)$.*
590 *Analogically for v .*

591 **Proof.** Only k^2 points can be covered only in *DIAG*, the rest has to be covered with
592 $VER \cup HOR$. Therefore every result must be at least $ALL_LINES - 2k^2\epsilon$, because only
593 $2k^2$ spaces of length ϵ can be uncovered in this axis.

594 Of course if h_{i,j,t_1} and $h_{i,j+1,t_2}$ are uncovered, then there must exist a segment in *HOR*
595 between h_{i,j,t_1}^R and $h_{i,j+1,t_2}^L$, so $math_h(t_1, t_2)$ must be true.

596 3.5. What is missing

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

598 Chapter 4

599 Geometric Set Cover with lines

600 4.1. Lines parallel to one of the axis

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

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

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

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

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

612 Non-weighted setting - Konig theorem + max matching

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

614 4.2. FPT for arbitrary lines

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

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

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

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

621 4.3. APX-completeness for arbitrary lines

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

625 Then for every edge in $(v, w) \in E(G)$ we put a point on crossing of lines for vertices v
626 and w . They are not pararell, so there exists exactly one such point and any other line don't
627 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.

4.4. 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.

4.5. 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.

Chapter 5

Geometric Set Cover with polygons

5.1. State of the art

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

Although with thin objects, even if we allow δ -expansion, the Set Cover with rectangles is APX-complete (for $\delta = 1/2$), it follows from APX-completeness for segments with δ -expansion in Section 3.2.

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

⁶⁶⁴ Chapter 6

⁶⁶⁵ Conclusions

666 Bibliography

- 667 [Har-Peled and Lee, 2009] Har-Peled, S. and Lee, M. (2009). Weighted geometric set cover
668 problems revisited. *Journal of Computational Geometry*, 3.
- 669 [Håstad, 2001] Håstad, J. (2001). Some optimal inapproximability results. *J. ACM*,
670 48(4):798–859.
- 671 [Li and Jin, 2015] Li, J. and Jin, Y. (2015). A PTAS for the weighted unit disk cover problem.
672 *CoRR*, abs/1502.04918.
- 673 [Marx, 2005] Marx, D. (2005). Efficient approximation schemes for geometric problems? In
674 Brodal, G. S. and Leonardi, S., editors, *Algorithms – ESA 2005*, pages 448–459, Berlin,
675 Heidelberg. Springer Berlin Heidelberg.
- 676 [Marx and Pilipczuk, 2015] Marx, D. and Pilipczuk, M. (2015). Optimal parameterized algo-
677 rithms for planar facility location problems using voronoi diagrams. *CoRR*, abs/1504.05476.