University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Katarzyna Kowalska
Student no. 371053

Approximation and Parametrized Algorithms for Segment Set Cover

Master's thesis
in COMPUTER SCIENCE

Supervisor:
dr Michał Pilipczuk
Instytut Informatyki

June 2020

Supervisor's statement

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

Date Supervisor's signature

Author's statement

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

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

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

Date Author's signature

Abstract

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

Keywords

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

Thesis domain (Socrates-Erasmus subject area codes)

11.3 Informatyka

Subject classification

D. Software
D.127. Blabalgorithms
D.127.6. Numerical blabalysis

Tytuł pracy w języku polskim

Algorytmy parametryzowania i trudność aproksymacji problemu pokrywania zbiorów odcinkami na płaszczyźnie

Contents

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 were 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 suffciently small $\epsilon > 0$, the problem does not admit $1+\epsilon$-approximation scheme) [refrencje].

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 segments 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 unweighed 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 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 $f : \mathcal{P} \to \mathbb{R}^+$, 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 2.2.1. A **polynomial-time approximation scheme (PTAS)** for a minimization problem Π is a family of algorithms $\mathcal{A}_\epsilon$ for every $\epsilon > 0$ such that $\mathcal{A}_\epsilon$ 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 $(1+\epsilon)opt(I)$, where $opt(I)$ is the weight of an optimal solution for I.

Definition 2.2.2. 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 2.3.1 (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 2.3.2 (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. $|\mathcal{R}| \leq |opt(I)|$.

TODO: Some text

Definition 2.3.3 (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. $(1 + \epsilon)|\mathcal{R}| \leq |opt(I)|$.

Chapter 3

Geometric Set Cover with segments

3.1. FPT for segments

3.1.1. Axis-parallel segments

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

We show $\mathcal{O}(2^k)$ branching algorithm. Let us take the point K which is the smallest under a lexicographic ordering on coordinates among points that are not covered yet. We need to cover K with some of the remaining segments.

We branch over choice of direction among the 2 axis-parallel directions. In this direction we greedly take the segment that covers the most points. As K was the smallest in lexicographical order, all points in $\mathcal{C}$ colinear with K in both axis-parallel directions are only on one side of K, because their coordinates are larger. Therefore segments covering K in this direction create monotone sequence of sets and we can greedily take one segment that covers superset of all of these segments.

TODO: Maybe split it into theorem + algorithm + explanation like in section 3.1.3

3.1.2. Segments in d directions

The same algorithm as described in the previous section, but we branch over d directions and it runs in complexity $\mathcal{O}(d^k)$.

3.1.3. Segments in arbitrary direction

Theorem 3.1.1. ***(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 $f(k) \cdot (nm)^c$ for some computable function f and constant c, 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 determins that the solution of size at most k doesn't exist.*

This theorem is proved by following lemmas.

Lemma 3.1.1. ***(Reduction).*** *Given a family $\mathcal{P}$ of n segments (in any direction) and a set of m points $\mathcal{C}$ for 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 $A, B \in \mathcal{P}$, we have $A \neq B \Rightarrow A \not\subseteq B$.*

Proof. Trivial. □

Lemma 3.1.2. *Given an instance of a problem, if there exists a line L with at least $k+1$ points on it, 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 $|\mathcal{A} \cap \mathcal{R}| \geq 1$.*

Proof. First we use Lemma 3.1.1.

Let us name points from $\mathcal{C}$ that lay on L, $x_1, x_2, \ldots x_t$ in the order they appear on the line.

Every segment that is not colinear 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 colinear with L.

Therefore we need to take one of the segments colinear with L that covers any of the points $x_1, x_2, \ldots x_k$. After using reduction from Lemma 3.1.1, there are at most k such segments that are distinct.

□

Proof of theorem 3.1.1.

Algorithm. First we use Lemma 3.1.1.

We present a recursive algorithm. Given an instance of the problem:

(1) If there exist a line with at least $k+1$ points, we branch over adding to the solution one of at most k possible segments from Lemma 3.1.2, name this segment S. Then we find a solution $\mathcal{R}$ for problem for points $\mathcal{C} - S$, segments $\mathcal{P} - \{S\}$ and parameter $k-1$ and 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 algorithm.

Correctness. Lemma 3.1.2 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 the optimal solution.

In (2) the answer is no, because every line covers no more than k points from $\mathcal{C}$, which implies that every segment from $\mathcal{P}$ covers at most k. 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.

Complexity. In leaves of branching (3) $|\mathcal{C}| < k^2$, so $|\mathcal{P}| < k^4$, because every segments can be uniquely identified by 2 extreme points it covers (by Lemma 3.1.1). Therefore there are $\binom{k^4}{k}$ possible solutions to check, each can be checked in time $O(k|\mathcal{C}|)$. Therefore (3) takes time $O(f(k))$.

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

Reduction from Lemma 3.1.1 can be implemented in $O(n^2 m)$.

Overall complexity is $O(n^2 m + nm \log(nm) \cdot f(k))$

□

3.2. 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 3.2.1. ***(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 3.2.1 implies the following.

Corollary 3.2.1. ***(rectangle set cover is APX-hard).*** *Unweighted geometric set cover with rectangles (even with 1/2-extension) is APX-hard.*

We prove Theorem 3.2.1 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.

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

Definition 3.2.1. 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 3.2.2. 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 3.2.1 we use hardness of approximation of MAX-(3,3)-SAT proved in [Håstad, 2001] and described in Theorem 3.2.2 below.

Definition 3.2.3 (α-satisfiable MAX-3SAT formula)**.** MAX-3SAT formula of size n is at most α-satisfiable, if every assignment of variables does not satisfy more than αn clauses.

Theorem 3.2.2. ***[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 3.2.2) 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 3.2.1.

Lemma 3.2.1. *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 3.2.1 in subsequent sections, but meanwhile let us prove Theorem 3.2.1 using Lemma 3.2.1 and Theorem 3.2.2.

TODO: This below can't use current template

Proof of Theorem 3.2.1. . 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 3.2.2, 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 3.2.1. 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 3.2.1 we have:

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

Therefore,

$$approx(I) \leq (1+\epsilon)opt(I) = 14n(1+\epsilon) = 14n + 14\epsilon \cdot n =$$

$$= 14n + (15\epsilon - \epsilon)n < 14n + \left(\frac{1}{8} - \epsilon\right)n = 15n - \left(\frac{7}{8} + \epsilon\right)n$$

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

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

From Lemma 3.2.1 we have:

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

Since a solution to I with $\frac{1}{2}$-extensions is also a solution without extentions, by Lemma 3.2.1 (3.), we have:

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

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

$\square$

3.2.2. Reduction construction

We show reduction from MAX-(3,3)-SAT problem to geometric set cover with segments parallel to axis. Moreover the instance of geometric set cover will be robust to 1/2-extensions (have the same optimal solution after 1/2-extension).

The construction will be composed of 2 types of gadgets: **VARIABLE-gadgets** and **CLAUSE-gadgets**. CLAUSE-gadgets would be constructed using two **OR-gadgets** connected together.

3.2.2.1. VARIABLE-gadget

VARIABLE-gadget is responsible for choosing the value of a variable in a CNF formula. It allows two minimal solutions and every minimal solution must use exactly one of the (c_i, g_i) and (f_i, h_i) segments. These two choices correspond to the two Boolean values of the variable.

Points. Define points:

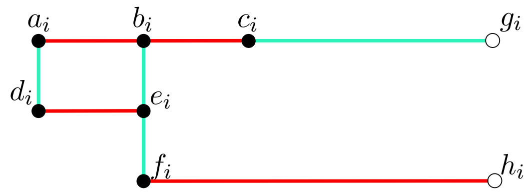

Figure 3.1: **VARIABLE-gadget.** We denote the set of points marked with black circles as C^i_{var}, and they need to be covered (are part of the set $\mathcal{C}$). Note that some of the points are not marked as black dots and exists only to name segments for further reference. We denote the set of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

With $L = 12n$:

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

Let us define:

$$C_{var} = \{a, b, c, d, e, f\}$$

and

$$C^i_{var} = C_{var} + (0, 4i)$$

Segments. Let us define:

$$X^i_{true} = \{(a_i, d_i), (b_i, f_i), (c_i, g_i)\}$$
$$X^i_{false} = \{(a_i, c_i), (d_i, e_i), (f_i, h_i)\}$$

$$P^i_{var} = X^i_{true} \cup X^i_{false}$$

Lemma 3.2.2. *For any $1 \leq i \leq n$, points C^i_{var} can be covered using 3 segments from P^i_{var}.*

Proof. We can use either set X^i_{true} or X^i_{false}. □

Lemma 3.2.3. *For any $1 \leq i \leq n$, points C^i_{var} can not be covered with less than 3 segments from P^i_{var}.*

Proof. No segment of P^i_{var} covers more than one point from $\{d_i, f_i, c_i\}$, therefore C^i_{var} can not be covered with less than 3 segments. □

Lemma 3.2.4. *If both segments (c_i, g_i) and (f_i, h_i) are chosen, then the covering the remaining points from C^i_{var} requires at least 2 different segments from P^i_{var}.*

Proof. No segment of P^i_{var} covers more than one point from $\{a_i, e_i\}$, therefore C^i_{var} - $\{c_i, f_i, g_i, h_i\}$ can not be covered with less than 2 segments. □

3.2.2.2. OR-gadget

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

$$(x \in R) \vee (y \in R) \iff result \in R$$

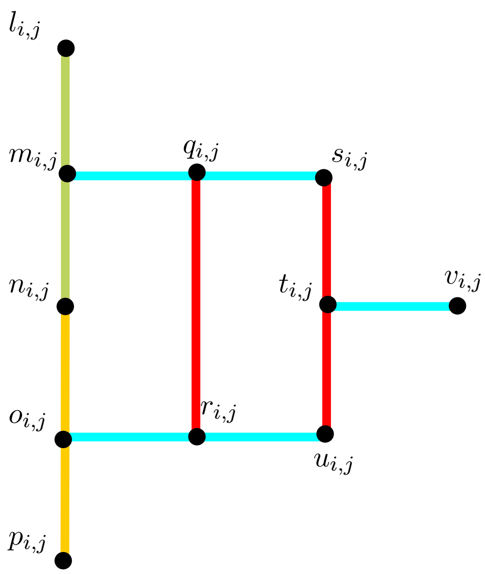

Figure 3.2: **OR-gadget.** We denote these point as $or\_gadget_{i,j}$. We denote set of red segments as $or^{false}_{i,j}$, set of blue segments as $or^{true}_{i,j}$, green and yellow segments as $or\_move\_variable_{i,j}$.

Points.

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

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

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

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

$$C\_or\_gadget_{i,j} = \{l_{i,j}, m_{i,j}, n_{i,j}, o_{i,j}, p_{i,j}, q_{i,j}, r_{i,j}, s_{i,j}, t_{i,j}, u_{i,j}\}$$

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

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

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

$$or\_move\_variable_{i,j} = \{(l_{i,j}, n_{i,j}), (n_{i,j}, p_{i,j})\}$$

Segments in OR-gadget:

$$P\_or\_gadget_{i,j} = or_{i,j}^{false} \cup or_{i,j}^{true} \cup or\_move\_variable_{i,j}$$

Lemma 3.2.5. *For any $1 \leq i \leq n, j \in \{0,1\}$ and $x \in \{l_{i,j}, p_{i,j}\}$ we can cover points in $C\_or\_gadget_{i,j} - \{x\} \cup \{v_{i,j}\}$ with 4 segments.*

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

Lemma 3.2.6. *For any $1 \leq i \leq n, j \in \{0,1\}$, we can cover points in $C\_or\_gadget_{i,j}$ with 4 segments from $P\_or\_gadget_{i,j}$.*

Proof. We can do that using $or\_move\_variable_{i,j}$ and $or_{i,j}^{false}$. □

3.2.2.3. CLAUSE-gadget

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

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

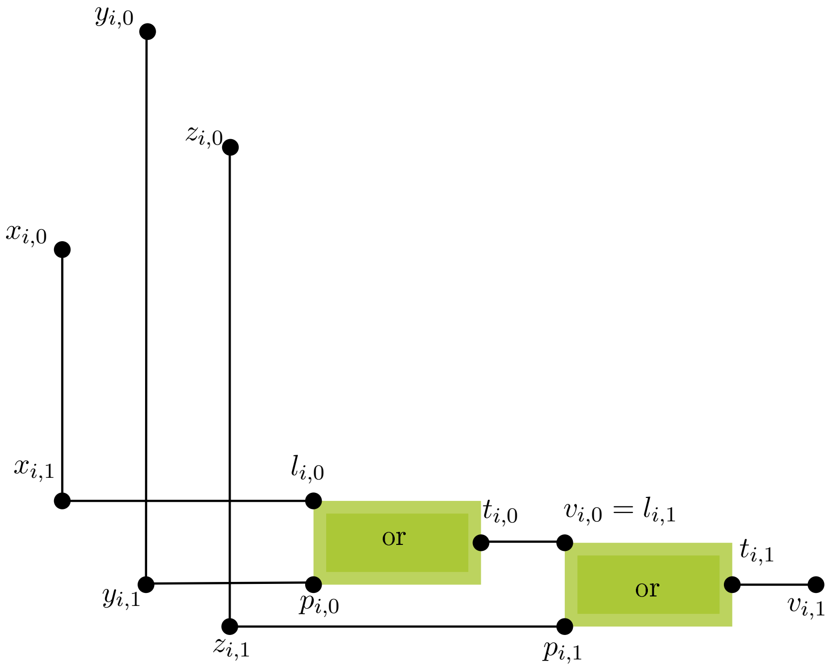

Figure 3.3: **CLAUSE-gadget.** We denote set of these points as C_clause_i. Every green rectangle is an OR-gadget. y-coordinates of $x_{i,0}$, $y_{i,0}$ and $z_{i,0}$ depend on the values of variables in the i-th clause.

Points. TODO: Rephrase it

Assuming clause $C_i = x_i \lor y_i \lor z_i$, function $idx(w)$ is returning index of the variable w, function $neg(w)$ is returning whether variable w is negated in a clause.

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

$$move\_variable_i = \{x_{i,j} : j \in \{0,1\}\} \cup \{y_{i,j} : j \in \{0,1\}\} \cup \{z_{i,j} : j \in \{0,1\}\}$$

$$C\_clause_i = move\_variable_i \cup C\_or\_gadget_{i,0} \cup C\_or\_gadget_{i,1} \cup \{v_{i,1}\}$$

Segments.

$$\begin{aligned} P\_clause_i \;\; = \;\; & \{(x_{i,0}, x_{i,1}), (y_{i,0}, y_{i,1}), (z_{i,0}, z_{i,1}), (x_{i,1}, l_{i,0}), (y_{i,1}, p_{i,0}), (z_{i,1}, p_{i,1}), \} \cup \\ & \cup P\_or\_gadget_{i,0} \cup P\_or\_gadget_{i,1} \end{aligned}$$

Lemma 3.2.7. *For any $1 \le i \le n$ and $a \in \{x_{i,0}, y_{i,0}, z_{i,0}\}$, points $C\_clause_i - \{a\}$ can be covered using 11 segments from P_clause_i.*

Proof. For $a = x_{i,0}$ (analogous proof for $y_{i,0}$): First we use Lemma 3.2.5 twice with excluded $x = l_{i,0}$ and $x = l_{i,1} = v_{i,0}$, resulting with 8 segments $or^{true}_{i,0} \cup or^{true}_{i,1}$ 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 $a = z_{0,i}$: Using Lemma 3.2.6 and Lemma 3.2.5 with $x = p_{i,1}$, resulting with 8 segments $or_{i,0}^{false} \cup or_{i,1}^{true}$ 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: $\{(x_{i,0}, x_{i,1}), (y_{i,0}, y_{i,1}), (z_{i,1}, p_{i,1})\}$. □

Lemma 3.2.8. *Points C_clause_i can be covered with 12 segments from P_clause_i.*

Proof. Using Lemma 3.2.6 twice we can cover $or\_gadget_{i,0}$ and $or\_gadget_{i,1}$ with 8 segments.

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

□

Lemma 3.2.9. *For any $1 \leq i \leq n$, points $C\_clause_i - \{x_{i,0}, y_{i,0}, z_{i,0}\}$ can not be covered using less than 11 segments from P_clause_i.*

All points C_clause_i can not be covered with less than 12 segments from P_clause_i.

Proof of no cover with less than 12 segments. There is independent set of 12 points in $C\_clause_i \supseteq \{x_{i,0}, y_{i,0}, z_{i,0}, l_{i,0}, p_{i,0}, q_{i,0}, u_{i,0}, v_{i,0} = l_{i,1}, p_{i,1}, q_{i,1}, u_{i,1}, v_{i,1}\}$. □

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

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

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

$$Y = or\_gadget_{i,0} - \{l_{i,0}, p_{i,0}\}$$
$$Z = or\_gadget_{i,1} - \{l_{i,1}, p_{i,1}\}$$

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

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

Therefore C_clause_i must be covered with at least $3 + 4 + 4 = 11$ segments. □

3.2.2.4. Summary

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

Lemma 3.2.10. ***Robustness to 1/2-extensions**. For every segment $s \in \mathcal{P}$, s and $s^{+1/2}$ cover the same points from $\mathcal{C}$.*

3.2.3. Summary of contruction

We define:

$$\mathcal{C} := \bigcup_{1 \leq i \leq n} C\_variable_i \cup C\_clause_i$$
$$\mathcal{P} := \bigcup_{1 \leq i \leq n} P\_variable_i \cup P\_clause_i$$

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 diferent gadget), so we can prove lemmas *locally*.

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

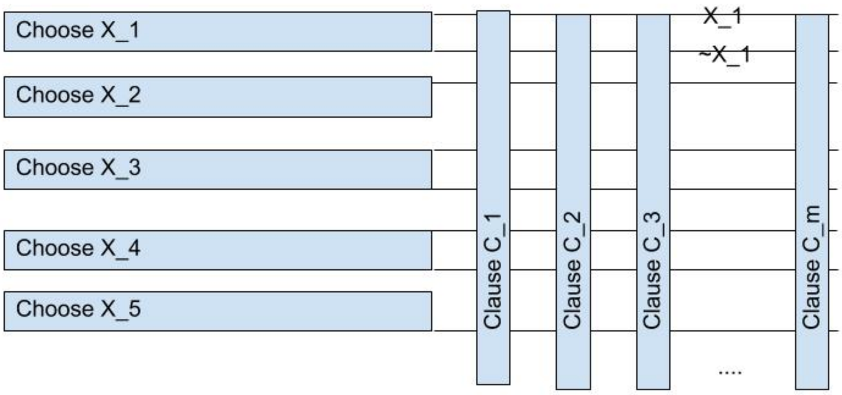

Figure 3.4: **General schema.**
General layout of VARIABLE-gadget and CLAUSE-gadget and how they interact with each other.
TODO: Rename Choose X to VARIABLE-gadget and Clause C to CLAUSE-gadget.

3.2.4. Construction lemmas and proof of Lemma 3.2.1

Lemma 3.2.11. *Given an instance S of MAX-(3,3)-SAT of size n with optimum solution satisfying k clauses $opt(S) = k$. Instance of geometric cover, constructed for S according to Lemma 3.2.1, can be solved with a solution of size $15n - k$.*

Proof. Let us name the assignments of the variables in the optimum solution of MAX-(3,3)-SAT instance y_1, $y_2 \ldots y_n$ and clauses c_1, $c_2 \ldots c_n$.

We cover every VARIABLE-gadget with solution described in Lemma 3.2.2, in the i-th gadget choosing the set of segments corresponding to the value of y_i. CLAUSE-gadgets that are satisfied are covered with set $clause_i^{true}$ described in Lemma 3.2.7 and unsatisfied with set $clause_i^{false}$ described in Lemma 3.2.8.

$$R_i = \begin{cases} x_i^{true} & \text{if } y_i \\ x_i^{false} & \text{if } \neg y_i \end{cases}$$
$$C_i = \begin{cases} clause_i^{true} & \text{if } c_i \text{ satisfied} \\ caluse_i^{false} & \text{if } c_i \text{ not satisfied} \end{cases}$$
$$\mathcal{R} = \bigcup_{i=1}^{n} \{R_i \cup C_i : 1 \leq i \leq n\}$$

This set covers all points form $\mathcal{C}$, because the smaller sets individually cover their corresponding gadgets (proved in respective lemmas).

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

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

.

□

Lemma 3.2.12. *Given an instance S of MAX-(3,3)-SAT of size n and a solution of size w to the instance of geometric cover, constructed for S according to Lemma 3.2.1. There exists a solution to MAX-(3,3)-SAT of size at least $15n - w$.*

Proof.

Variables We need to use at least 3 segments to cover VARIABLE-gadget (Lemma 3.2.3). If we have chosen both segments (c_i, g_i) and (f_i, h_i), then we have used at least 4 segments (Lemma 3.2.4).

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

If we chose at most one of the segments (c_i, g_i) and (f_i, h_i), choose the corresponding variable value to the solution. If we chose both segments, choose the value that appears in most clauses. Every variable is in exactly 3 clauses, so one value appears in at least 2 of them. If we have chosen none of the segments, set value to false.

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

To cover $\bigcup_{1 \leq i \leq n} C_{var}^i$ we have used at least $3n + a$ segments, where a is the number of i such that we have chosen both values (c_i, g_i) and (f_i, h_i).

Clauses For a clause $C_i = x \vee y \vee z$, we need to use at least 11 segments to cover $C\_clause_i - \{x, y, z\}$ in CLAUSE-gadget (Lemma 3.2.9).

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

Moreover, if all of the points $\{x, y, z\}$ are not covered by the segments from P_{var}^i, with at least 12 segments by Lemma 3.2.9.

We covered CLAUSE-gadget with at least 11 or at least 12 segments: $|\bigcap\limits_{i=1}^{n} P\_clause_i \cap \mathcal{R}| \geq 11n + b$, where b is the number of clauses where none of the segments covering the points x, y, z were chosen in P_{var}^j.

Satisfied clauses in chosen variables assignment Clauses for which none of the segments covering the points x, y, z were chosen in P_{var}^j, are not satisfied in our variables assignment, but not all clauses that cover one of these points with segment in P_{var}^j are satisfied.

Let us look at such equation and cases of choosing variable value in equation (3.1).

If only one of the segments (c_i, g_i) and (f_i, h_i) are chosen in P_var^i, then every clause that uses variable x_i with value that we chose is satisfied.

If we chose neither (c_i, g_i) or (f_i, h_i), then every clause that uses variable x_i as $false$ is satisfied. If the CLAUSE-gadget was covered by 11 segments instead of 12, then some other variable had to have its variable value segment covered.

If we chose both (c_i, g_i) and (f_i, h_i), then there might exist one clause that was covered by 11 segments instead of 12, but the clause is not satisfied with the value that we set to x_i. This happens because adding both (c_i, g_i) and (f_i, h_i) to the solution makes CLAUSE-gadgets, that use x_i with both values, possible to cover with 11 segments instead of 12, but we set value of x_i to the one used in majority of clauses, so there exists at most one that is covered with 11 segments and not satisfied. We had a such variables, so there are at most a clauses that are covered with 11 segments and not satisfied.

So in the solution to this MAX-(3,3)-SAT instance that we have shown, there are at most $a + b$ unsatsfied clauses.

Conclusions We proved that given a solution of size w we have the variables assignment that satisfies at least $n - (a + b)$ clauses of S. At last we prove that $n - (a + b) \geq 15n - w$.

$$w \geq 3(n - a) + 4a + 11(n - b) + 12b = 3n + a + 11n + b = 14n + a + b$$

$$15n - w \leq 15n - 14n - a - b = n - (a + b)$$

□

Proof of Lemma 3.2.1. Given an instance S of MAX-(3,3)-SAT of size n with optimum solution satisfying k clauses. Let us construct an instance of geometric cover for S, constructed in aforementioned manner and name it I.

Given the Lemma 3.2.11, we know the optimum solution for I has size at most $15n - k$. Since k is the optimum solution for S, then according to Lemma 3.2.12 there does not exist any solution for I with size less than $15n - k$. □

3.3. Weighted segments

3.3.1. FPT for weighted segments with δ-extensions

Theorem 3.3.1. ***(FPT for weighted segment cover with δ-extensions)***. *There exists an algorithm that given a family $\mathcal{P}$ of n weighted segments (in any direction), a set of m points $\mathcal{C}$ and a parameter k, runs in time $f(k) \cdot (nm)^c$ for some computable function f and constant c, and outputs a subfamily $\mathcal{R} \subseteq \mathcal{P}$ such that $|\mathcal{R}| \leq k$ and $\mathcal{R}^{+\delta}$ covers all points in $\mathcal{C}$.*

To solve this problem we will introduce kernel for slightly different problem: Weighted segment cover of points and segments. In shortcut: WSCPS.

Lemma 3.3.1. ***(Algorithm for kernel of WSCPS)***. *There exists an algorithm that given a family $\mathcal{P}$ of n weighted segments (in any direction), a set of m_1 points $\mathcal{C}_1$ and m_2 segments $\mathcal{C}_2$ and a parameter k, runs in time $f(k) \cdot g(m_1, m_2) \cdot n^c$ for some computable functions f, g and constant c, and outputs a subfamily sol $\subseteq \mathcal{P}$ such that $|\mathcal{R}| \leq k$ and $\mathcal{R}$ covers all points in $\mathcal{C}_1$ and all segments in $\mathcal{C}_2$.*

Proof Only sketch for now.

We can compute dynamic programming $dp(A, B, z)$ – the best cost to cover at least whole segment A, B using at most z segments. A, B are all interesting points – ends of any segment given on the input or points given on the input. We can compute it in polynomial time.

Then we can create a new double weighted set (original weight, number of used segments from $\mathcal{P}$) – $\mathcal{P}_2$ that has only segments which never cover partially any segment from $\mathcal{C}_2$ (covers the whole segment or doesn't cover at all). In such $\mathcal{P}_2$ we can find solution $\mathcal{R}$ where any 2 segments have empty intersection (don't cover each other and don't meet at the ends). Because if we had such solution, we can merge these two segments and such segment there's also in $\mathcal{P}_2$.

In that case we can find kernel of $\mathcal{P}_2$ of size $k \cdot (m_1 + 2m_2)^2$, because we only need to take the best weight covering some subset of $\mathcal{C}_1 \cup \mathcal{C}_2$.

Lemma 3.3.2. ***Kernel in WSCPS***. *TODO: formulate it properly*

For segment cover, there is a kernel of size $f(k)$ in WSCPS.

Claim 3.3.1. *If there are more than k lines with at least $k+1$ points on them, then they can't be covered with k segments.*

Claim 3.3.2. *If there is more than k^2 points that don't lie on any line with more than k points on it, then they can't be covered with k segments.*

Claim 3.3.3. *For every long line L (with more than k points on them) we can choose $f(k)$ points on them, that if we cover all of these points with at most k segments, then the rest of the points with δ-extensions will be covered by segments in the direction of line L.*

Proof of Lemma 3.3.2. After applying the previous lemmas, we have at most $k^2 + k \cdot f(k)$ points that can be covered in any direction and for the rest of the points we can draw at most $k \cdot f(k)$ segments along their respective long lines that have to be covered by segments after δ-extensions.

Then we extend every available segment by δ-extension and we achieve the kernel in WSCPS for this instance of problem.

Lemma 3.3.3. *If all the points are covered with k segments and the biggest $2(1 + 1/\delta)^{k+1}$ spaces between points are filled, the whole segment is filled after δ-extensions of these segments.*

Proof. Let's name the $2(1+1/\delta)^{k+1}$-st biggest space between points as y. We have guarantee that all segements of length $x > y$ are covered without δ-extensions.

Let's take one space between points that is not covered before δ-extension and we will prove it will be covered after δ-extensions. Let's assume it isn't.

This space has length x. Since it's uncovered, $x \leq y$.

Let's take side where the sum of lengths of segments covering the points is greater (left or right). Without loss of generality, let us assume it's right.

There are at most k segments to the right of this space between points. Name their lengths $l_1, l_2 \dots l_k$. If the point is covered in the other direction, the segment is degenerated to the point and $l_i = 0$. Name the space between endpoints of l_i and l_{i+1} – x_i. Of course, x_i is uncovered space between two points, therefore $x_i \leq y$.

TUTAJ BEDZIE PEWNIE RYSUNEK Z TYMI SUPER RZECZAMI DO PRZERW

Let's write equations meaning that i-th segment doesn't cover space x after δ-expansion.

$$l_1\delta < x \leq y \Rightarrow l_1 < y/\delta$$

$$l_2\delta < x + l_1 + x_1 < 2y + y/\delta \Rightarrow l_2 < 2y/\delta + y/\delta^2$$

$$l_3\delta < x + l_1 + x_1 + l_2 + x_2 < 3y + 3y/\delta + y/\delta^2 \Rightarrow l_3 < 3y/\delta + 3y/\delta^2 + y/\delta^3$$

From this we can "guess" induction $l_i < y((1 + 1/\delta)^i - 1)$

Trivially for $l_1 < y/\delta$.

Assume that for all $j < i$:

$$l_j < y((1 + 1/\delta)^j - 1)$$

.

$l_i\delta < x + \sum_{j=1}^{i-1}(l_j + x_j) < iy \sum_{j=1}^{i-1} l_j < iy + \sum j = 1^{i-1} y((1 + 1/\delta)^j - 1) = iy - (i - 1)y + \sum j = 1^{i-1} y(1 + 1/\delta)^j = y(1 + \sum_{j=1}^{i-1}(1 + 1/\delta)^j) = y(2 + \sum_{j=1}^{i-1}(1 + 1/\delta)^j - 1) = y(\sum_{j=0}^{i-1}(1+1/\delta)^j - 1) = y((1+1/\delta)^i/(1-(1+1/\delta))-1) = y((1+1/\delta)^i\delta - 1) < y((1+1/\delta)^i\delta - \delta)$

Of course we also know that (since we have chosen the side with greater sum of the width of segments):

$$\sum_{i=1}^{k} l_i \geq 1/2 \cdot y \cdot 2(1 + 1/\delta)^{k+1} = y \cdot (1 + 1/\delta)^{k+1}$$

But $\sum_{i=1}^{k} l_i < \sum_{i=1}^{k} y((1 + 1/\delta)^i - 1) = y((1 + 1/\delta)^{k+1}/(1 - (1 + 1/\delta)) - k) = y((1 + 1/\delta)^{k+1}\delta - k) < y(1 + 1/\delta)^{k+1}$

Therefore the space must have been covered after δ-expansions.

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

Theorem 3.3.2. ***W[1]-completeness for weighted segments in 3 directions.*** *Consider the problem of covering a set $\mathcal{C}$ of points by selecting k axis-pararell 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 gird 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 $a_1 \dots a_{n^2}$.

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

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

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.

$$hor_{i,j,t_1,t_2} = (h^R_{i,j,t_1}, h^L_{i,j+1,t_2})$$

$$ver_{i,j,t_1,t_2} = (v^D_{i,j,t_1}, v^U_{i,j+1,t_2})$$

$$horbeg_{i,t} = (h^L_{i,1,1}, h^L_{i,1,t})$$

$$horend_{i,t} = (h^R_{i,n,t}, h^R_{i,n,n^2})$$

$$verbeg_{i,t} = (v^U_{i,1,1}, v^U_{i,1,t})$$

$$verend_{i,t} = (v^D_{i,n,t}, v^D_{i,n,n^2})$$

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

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

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

TODO: explain that these segments are in fact diagonal

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

Lemma 3.3.4. *If there exists solution for grid tiling, then there exists solution for our construction using* $2(k+1)k + k^2$ *segments with weight exactly* $2k \cdot (k(n^2+1) - 2 - 2\epsilon(k-1))$.

Claim 3.3.4. *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*

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

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

Proof. TODO: jakiś prosty z definicji

Lemma 3.3.5. *If there exists solution for our construction using* $2(k+1)k + k^2$ *segments with weight exactly* $2k \cdot (k(n^2+1) - 2 - 2\epsilon(k-1))$*, then there exists a solution for grid tiling*

Proof. This follows from Lemma 3.3.6, because we just take which points are covered with $DIAG$.

Claim 3.3.5. *Points* p^L, p^R, p^U, p^D *cannot be covered with* $DIAG$.

Claim 3.3.6. *Points in* $H \cup \{p^L : p \in H\} \cup \{p^R : p \in H\}$ *cannot be covered with* VER.
Points in $V \cup \{p^U : p \in V\} \cup \{p^D : p \in V\}$ *cannot be covered with* HOR.

Claim 3.3.7. *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* $DIAG$*, then some spaces between neighbouring points were covered twice.*

Claim 3.3.8. *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 with* $DIAG$*, then one of them had to be also covered with a segment from* HOR *(*VER*).*

Proof. Point v^L_{i,j,t_2} had to be covered with VER from Claims 3.3.5 and 3.3.6. And every segment in VER covering v^L_{i,j,t_2}, covers also v^L_{i,j,t_1}.

Lemma 3.3.6. *If there exists solution for our construction with weight at most (exactly)* $2k \cdot (k(n^2+1) - 2 - 2\epsilon(k-1))$*, then for every* i, j *there must be exactly one* t *such that* $h_{i,j,t}$ *(*$v_{i,j,t}$*) is covered with* $DIAG$ *and moreover if* h_{i,j,t_1} *and* $h_{i,j+1,t_2}$ *are uncovered, then* $math_h(t_1, t_2)$*. Analogically for* v*.*

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

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 between h^R_{i,j,t_1} and $h^L_{i,j+1,t_2}$, so $math_h(t_1, t_2)$ must be true.

3.3.3. What is missing

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

Chapter 4

Geometric Set Cover with lines

4.1. 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)

4.2. 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.

4.3. 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 in 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.

Unfortunaly 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].

Chapter 6

Conclusions

Bibliography

[Har-Peled and Lee, 2009] Har-Peled, S. and Lee, M. (2009). Weighted geometric set cover problems revisited. *Journal of Computational Geometry*, 3.

[Håstad, 2001] Håstad, J. (2001). Some optimal inapproximability results. *J. ACM*, 48(4):798–859.

[Li and Jin, 2015] Li, J. and Jin, Y. (2015). A PTAS for the weighted unit disk cover problem. *CoRR*, abs/1502.04918.

[Marx, 2005] Marx, D. (2005). Efficient approximation schemes for geometric problems? In Brodal, G. S. and Leonardi, S., editors, *Algorithms – ESA 2005*, pages 448–459, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Marx and Pilipczuk, 2015] Marx, D. and Pilipczuk, M. (2015). Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. *CoRR*, abs/1504.05476.