Better Diameter Algorithms for Bounded VC-Dimension Graphs and Geometric Intersection Graphs

Lech Duraj
Faculty of Mathematics and Computer Science, Jagiellonian University in Kraków, Poland

Filip Konieczny
Faculty of Mathematics and Computer Science, Jagiellonian University in Kraków, Poland

Krzysztof Potępa
Faculty of Mathematics and Computer Science, Jagiellonian University in Kraków, Poland

Abstract

We develop a framework for algorithms finding the diameter in graphs of bounded distance Vapnik-Chervonenkis dimension, in (parameterized) subquadratic time complexity. The class of bounded distance VC-dimension graphs is wide, including, e.g. all minor-free graphs.

We build on the work of Ducoffe et al. [SODA'20, SIGCOMP'22], improving their technique. With our approach the algorithms become simpler and faster, working in $\mathcal{O}\big(k \cdot n^{1-1/d} \cdot m \cdot \mathrm{polylog}(n)\big)$ time complexity for the graph on n vertices and m edges, where k is the diameter and d is the distance VC-dimension of the graph. Furthermore, it allows us to use the improved technique in more general setting. In particular, we use this framework for geometric intersection graphs, i.e. graphs where vertices are identical geometric objects on a plane and the adjacency is defined by intersection. Applying our approach for these graphs, we partially answer a question posed by Bringmann et al. [SoCG'22], finding an $\mathcal{O}\big(n^{7/4} \cdot \mathrm{polylog}(n)\big)$ parameterized diameter algorithm for unit square intersection graph of size n, as well as a more general algorithm for convex polygon intersection graphs.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Computational geometry

Keywords and phrases Graph Diameter, Geometric Intersection Graphs, Vapnik-Chervonenkis Dimension

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.51

Related Version *Full Version*: `https://arxiv.org/abs/2307.08162`

Funding All authors were supported by National Science Center of Poland grant 2019/34/E/ST6/00443.
Filip Konieczny: During the preparation of this article, Filip Konieczny was a participant of the tutoring programme under the Excellence Initiative at the Jagiellonian University.

Acknowledgements We would like to thank the reviewers for helping to improve our paper with their suggestions.

1 Introduction

The *diameter* of a graph is the maximum possible distance between a pair of vertices. It is believed to be an important graph parameter and as such, it has been extensively studied. Formally, the DIAMETER and k-DIAMETER problems are defined as follows[1]:

[1] In this work, we assume graphs to be unweighted and undirected.

- DIAMETER: Given a graph $G = (V, E)$, calculate $\mathrm{diam}(G) := \max_{u,v \in V} \mathrm{dist}(u, v)$, where $\mathrm{dist}(u, v)$ is the shortest possible length of any path between u and v;
- k-DIAMETER: Given a graph $G = (V, E)$ and $k \in \mathbb{Z}$, determine whether $\mathrm{diam}(G) \leq k$.

Both DIAMETER and k-DIAMETER are easy to solve in $\mathcal{O}(nm)$ time complexity, where $n = |V|$, $m = |E|$, by simply invoking BFS from every vertex. This straightforward algorithm, however, turns out to be notoriously hard to improve in terms of time complexity. In 2013, Roditty and Vassilevska-Williams showed [28] that any algorithm solving 2-DIAMETER in $\mathcal{O}\big(m^{2-\varepsilon}\big)$ time complexity would imply the existence of a $(2-\delta)^n$ time algorithm for SAT, thus refuting Strong Exponential Time Hypothesis (SETH, [24]). Although conditional, it is an argument for the existence of a quadratic complexity barrier. Furthermore, the hardness of 2-DIAMETER implies that approximating the diameter with ratio better than 3/2 in subquadratic time would refute SETH as well. But even if we assume SETH to be true, there is still a lot of open questions about diameter. One long line of research deals with subquadratic approximation in general graphs, and trade-offs between complexity and approximation ratio [28, 12, 7, 25, 26, 15, 16, 3, 1].

Another family of questions arises from considering the diameter problem for some restricted graph classes ([2, 14, 27, 21]. A notable example is the case of planar graphs: the first subquadratic algorithm was found by Cabello [6], and the fastest currently known works in $\widetilde{\mathcal{O}}\big(n^{5/3}\big)$ time[2] and is due to Gawrychowski et al. [22]. Similar problems arise from considering *geometric intersection graphs*: we take a family of objects in $\mathbb{R}^k$, name them the vertices of our graph, and define an edge between a pair of objects to exist if and only if they intersect. There is a natural interpretation of a diameter problem for these graphs, especially if the objects are unit balls or axis-aligned unit squares on the plane: the middle point of each object is a communication node, and the object itself represents its maximal range of communication. The diameter of the graph is the maximal number of hops needed for any two nodes to successfully communicate. Observe that the resulting graph on n objects can easily have $\Theta(n^2)$ edges, as well as very large cliques. This makes a subquadratic algorithm somewhat more tricky, as we cannot ever list the edges of this graph explicitly, but instead we have to rely on its geometric representation. Geometric intersection graphs have also been studied in terms of fine-grained complexity [8, 9], but there were relatively few subquadratic breakthroughs for the diameter problem. A recent paper by Bringmann et al. [5], proved (among other results) that:

- neither the intersection graph of axis-parallel unit cubes nor unit balls in $\mathbb{R}^3$ admits a subquadratic diameter algorithm under SETH;
- the intersection graph of axis-parallel unit cubes in $\mathbb{R}^{12}$ does not admit, under the Hyperclique Hypothesis, a subquadratic algorithm for 2-DIAMETER;
- for the intersection graph of axis-parallel unit squares in $\mathbb{R}^2$ there is an algorithm with $\mathcal{O}(n \log n)$ time complexity for 2-DIAMETER.

Which other graph classes are non-trivial to consider in this setting? Some good choices are, for example, K_t-minor-free graphs, as they forbid using the counterexample from [28]. Ducoffe, Habib and Viennot [18, 19] proposed a more general class of graphs to consider: the ones with bounded *distance Vapnik-Chervonenkis dimension*, or *distance VC-dimension* for short. We formally define it in section 2, but roughly speaking, graph has distance VC-dimension bounded by d, if for every subset $A \subseteq V$ with $|A| > d$ there exists $A' \subseteq A$ which cannot be expressed as a projection of a ball, i.e. in the form $A' = \{x \in A : dist(x, v) \leq k\}$

[2] The $\widetilde{\mathcal{O}}()$ notation ignores logarithmic factors, i.e. $\widetilde{\mathcal{O}}(f(n))$ means $\mathcal{O}(f(n) \cdot \mathrm{polylog}(f(n)))$.

for some $v \in V$ and $k \in \mathbb{Z}$. The class of bounded distance VC-dimension graphs includes in particular minor-free graphs (with planar graphs), interval graphs, and also geometric intersection graphs. In their work, the authors of [19] showed a number of important results tying diameter finding to distance VC-dimension, in particular:

- a subquadratic algorithm for k-Diameter, working in $\widetilde{\mathcal{O}}\big(kn^{1-\varepsilon_d}m\big)$ time complexity, where $\varepsilon_d \sim \frac{1}{2^d \mathrm{poly}(d)}$ is some (small) constant[3];
- a subquadratic algorithm for Diameter, for graphs with bounded VC-dimension which additionally admit sublinear separators (e.g. minor-closed graphs).

While preparing this version of our paper, we discovered an independent work by Hsien-Chih Chang, Jie Gao and Hung Le [10]. Their main result is a subquadratic algorithm which computes an additive-constant approximate (+2) diameter of a geometric intersection graph for any family of *pseudo-disks* (the pseudo-disks are shapes bounded by a Jordan curve with a property that two such boundaries can have at most two intersection points; in particular, the graphs considered in this paper fit into that category). In Section 6 we discuss how our contributions are related.

1.1 Our contribution and paper structure

An inspiration for this paper was to answer the open questions posed in [5]; especially, to find a (parameterized) subquadratic algorithm for some geometric intersection graphs. In the most appealing cases of planar unit disk and unit square intersection graphs, it is not hard to prove that both these classes have their distance VC-dimension bounded by 4. Therefore, algorithms from [19] could in theory be applied to them, but it is impossible to do it directly, as those algorithms work only for explicitly-given sparse graphs. Therefore, we need to refine this algorithm to work in our setting.

The core idea of [19] is to find a spanning path of a graph $G = (V, E)$ with a low *stabbing number*, i.e. an order $v_1, \ldots, v_n$ on the vertices of the graph such that for every $v \in V$, $k \in \mathbb{Z}$, every ball $N^k[v]$ can be expressed as a sum of $\mathcal{O}\big(n^{1-\varepsilon}\big)$ intervals $(v_x, v_{x+1}, \ldots, v_y)$. The existence of such a path is in turn based on the results of Chazelle and Welzl [11], who provide a Monte Carlo polynomial algorithm for finding such a path. The authors of [19] use this algorithm as a subroutine ("black-box"), employing a neat trick to bring down its polynomial complexity to a subquadratic one.

Interestingly, the Chazelle-Welzl subroutine uses a technique similar to the main construction of [19] – in particular, the notion of ε-nets, first introduced in [23]. In this paper we show that these two constructions can be, in a natural way, replaced by only one argument. This requires going back on the basic definitions, in particular relaxing the conditions on the stabbing number, as well as different complexity analysis. We are, however, rewarded with a simpler and more straightforward algorithm, naturally working in subquadratic time. Furthermore, this also brings down the time complexity of the algorithm, and opens new possibilities of its generalization.

The high-level concept is as follows: for any j and for a vertex v let us denote by $N^j[v]$ the j-neighbourhood of v, i.e. all vertices reachable from v via at most j edges. We find a particular order $v_1, \ldots, v_n$ on all the vertices such that for every i the neighbourhoods $N^j[v_i]$ and $N^j[v_{i+1}]$ differ relatively little – to be precise, the total size of the difference sets $N^j[v_i] \triangle N^j[v_{i+1}]$ is subquadratic. This order can be found for every j with a randomized

[3] In fact, the 2^d factor in [19] is due to considering directed graphs and other technicalities; we believe that the authors could instead claim $\mathcal{O}\big(d^2\big)$, albeit with a large multiplicative constant.

algorithm, using the concept of ε-*nets*, and it allows us to encode all the j-neighbourhoods in subquadratic space. It is now enough to devise a way to compute this encoding also in subquadratic time; similarly to [19], we do it incrementally, going from all $N_{j-1}[v]$ sets to all $N_j[v]$ sets. We need, however, different constructions for general sparse graphs (when trying to improve [19]) and for implicitly given graphs (like geometric intersections). For the latter, we show that the key ingredient is a data structure, working on vertex subsets of our graph, allowing two particular operations: expanding a subset and computing the symmetric difference of two stored subsets. We devise such a data structure for axis-aligned unit-square graphs, and then generalize it to any convex polygons. Our structure is based on *persistent segment trees*, but to our knowledge, it has not been considered before in this form.

To sum up, we claim the following results:

- There is a randomized Las Vegas algorithm which, for any graph $G = (V, E)$ of distance VC-dimension at most d, solves k-DIAMETER in $\widetilde{\mathcal{O}}\big(k \cdot n^{1-1/d} \cdot m\big)$ time complexity (Section 3, Theorem 16);
- The algorithm above can be adapted to any class of implicitly given graphs, if provided an appropriate data structure, working on the graph's neighbour lists (Section 4, Theorem 18);
- In particular, for the axis-aligned unit square intersection graphs, there is a Monte Carlo algorithm solving k-DIAMETER in $\widetilde{\mathcal{O}}\big(k \cdot n^{7/4}\big)$ time complexity (Section 5, Theorem 20a);
- This algorithm can be generalized to any convex polygon intersection graphs, with an additional multiplicative constant depending on the polygon's number of sides (Section 5, Theorem 20b).

For the general graph algorithm, the new time complexity $\widetilde{\mathcal{O}}\big(k \cdot n^{1-1/d} \cdot m\big)$ is brought down from $\widetilde{\mathcal{O}}\Big(k \cdot n^{1-\frac{1}{2^{d \, poly(d)}}} \cdot m\Big)$ previously achieved in [19]. This is a more practical complexity, and we (tentatively) conjecture that this bound might be a tight one for the class of K_d-minor free graphs, or at least for the class of graphs of distance VC-dimension bounded by d.

As for the paper structure, Section 2 of this paper introduces the most important concepts, such as (distance) VC-dimension, ε-nets and related theorems. In Section 3 we introduce the main tools for constructing all the fast algorithms: the low-difference orders on graph vertices, and use them to improve the results for general sparse graphs. In Section 4 we show how to use these tools in the case of implicitly given graphs. Finally, in Section 5 we apply all these concepts to achieve the original goal – a parameterized subquadratic algorithm for unit-square graphs and then for general convex polygon intersection graphs.

2 Preliminaries

2.1 Graphs, neighbourhoods and diameters

We assume that the reader is familiar with the notion of graphs, paths and distances. Throughout the paper, all graphs are undirected and unweighted, as well as connected. We also use the same notation for most graphs: if $G = (V, E)$ is a graph, then let $n = |V|$ and $m = |E|$.

Given a graph $G = (V, E)$ and $S \subseteq V$ let $N[S]$ denote vertices in the (closed) neighbourhood of S, i.e. vertices belonging to S or having a neighbour in S. If $v \in V$ we let $N[v] = N(\{v\})$. We also introduce the notion of k-neighbourhood for $k \geq 0$, denoted recursively by $N^0[S] = S$, $N^k[S] = N[N^{k-1}[S]]$, which is the set of vertices with distance at most k to any vertex in S. As stated before, the diameter of G is $\operatorname{diam}(G) = \max_{u,v \in V} \operatorname{dist}(u, v)$. It is easy to see that the graph has diameter at most k if and only if for every $v \in V$ we have $N^k[v] = V$.

2.2 Hypergraphs and VC-dimension

A *hypergraph* is a pair $(X, \mathcal{R})$, where X is the set of *vertices* and $\mathcal{R} \subseteq \mathcal{P}(X)$ is a family of subsets of X, the *hyperedges*. Some natural examples of hypergraphs, which are most important for this paper, come from graph neighbourhoods. If $G = (V, E)$ is a graph, then:

- For $k \in \mathbb{Z}$, we define $\mathcal{N}^k(G) = \{N^k[v] : v \in V\}$ as the family of all possible k-neighbourhoods. The hypergraph $(V, \mathcal{N}^k(G))$ is the *k-distance hypergraph* of G;
- For the family of all balls $\mathcal{B}(G) = \bigcup_{k \geq 0} \mathcal{N}^k(G)$, we will call the hypergraph $(V, \mathcal{B}(G))$ the *ball hypergraph* of G.

As mentioned in the introduction, the key concept needed for our results is the *Vapnik–Chervonenkis dimension* [31] of a hypergraph $(X, \mathcal{R})$. It is defined as follows:

▶ **Definition 1.** *A hypergraph $(X, \mathcal{R})$ shatters a subset $Y \subseteq X$ if for every $Z \subseteq Y$ there exists $R \in \mathcal{R}$ such that $Z = R \cap Y$. In other words $|\{R \cap Y \mid R \in \mathcal{R}\}| = 2^{|Y|}$. The* Vapnik-Chervonenkis dimension *(or* VC-dimension*) of a hypergraph is the maximum size of a shattered subset.*

The following theorems recall some well-established properties of hypergraphs with bounded VC-dimension. The first one deals with VC-dimension of sub-hypergraphs and projection hypergraphs, the other one (Sauer-Shelah-Perles Lemma) bounds the number of hyperedges in terms of vertices and the VC-dimension. Most of our complexity bounds throughout the paper stem from this lemma.

▶ **Theorem 2.** *Let $(X, \mathcal{R})$ be a hypergraph where $|X| = n$, and let its VC-dimension be bounded by d. Then:*

1. *If $\mathcal{R}' \subseteq \mathcal{R}$, then hypergraph $(X, \mathcal{R}')$ also has VC-dimension bounded by d,*
2. *If $Y \subseteq X$, then hypergraph $(Y, \{Y \cap R \mid R \in \mathcal{R}\})$ also has VC-dimension bounded by d.*

▶ **Theorem 3** (Sauer-Shelah-Perles Lemma, [29, 30])**.** *For every integer d, there exists a constant $\beta = \beta(d)$ such that every hypergraph $(X, \mathcal{R})$ of VC-dimension at most d satisfies $|\mathcal{R}| \leq \beta \cdot |X|^d$.*

▶ **Corollary 4.** *For every integer d, there exists a constant $\beta = \beta(d)$ such that for every hypergraph $(X, \mathcal{R})$ of VC-dimension at most d and for every $S \subset X$, the cardinality of $\{S \cap R \mid R \in \mathcal{R}\}$ is at most $\beta |S|^d$.*

Let $(X, \mathcal{R})$, $(X, \mathcal{R}')$ be two hypergraphs on the same underlying set X. Suppose that both of them have VC-dimension d. By $\triangle$ we denote the *symmetric difference* operator on sets, i.e. $A \triangle B := (A \setminus B) \cup (B \setminus A)$. An important issue for us is bounding the VC-dimension of the hypergraph $(X, \{R \triangle R' \mid R \in \mathcal{R}, R' \in \mathcal{R}'\})$. The following lemma provides a bound of $\mathcal{O}(d \log d)$. It works for any operator $\circ$ on set such that intersection distributes over $\circ$ (i.e. $A \cap (B \circ B') = (A \cap B) \circ (A \cap B')$ for any sets A, B, B'; the union, intersection, set difference and symmetric difference operators all have this property. It is partially based on a similar lemma in [20], see the full version of the paper for more details and the proof.

▶ **Lemma 5.** *Let $(X, \mathcal{R})$, $(X, \mathcal{R}')$ be hypergraphs with VC-dimension not greater than d, and let $\circ : \mathcal{P}(X) \times \mathcal{P}(X) \to \mathcal{P}(X)$ be a binary set operator such that intersection distributes over $\circ$. Then the VC-dimension of $(X, \mathcal{R}^*)$, where $\mathcal{R}^* = \{R \circ R' \mid R \in \mathcal{R}, R' \in \mathcal{R}'\}$ is $\mathcal{O}(d \log d)$.*

Let us now define another one of this paper's central concepts, linking the notion of VC-dimension with graph diameters: the *distance VC-dimension* of a graph $G = (V, E)$.

▶ **Definition 6.** Distance VC-dimension *of a graph $G = (V, E)$ is the VC-dimension of its ball hypergraph, i.e. the hypergraph $(V, \mathcal{B}(G))$.*

We assume throughout the paper that we only consider graphs with distance VC-dimension at least 2, as there are no non-trivial connected graphs with distance VC-dimension 1. Finally, observe that by Theorem 2, if distance VC-dimension of a graph is bounded by some integer d, then for every $k \in \mathbb{N}$, the VC-dimension of $(V, \mathcal{N}^k(G))$ is also bounded by d.

Among others, the following classes of graphs have bounded distance VC-dimension: interval graphs, K_t-minor free graphs, and in general any minor-closed class of graphs [4, 13]. The next section is devoted to establishing similar bounds for geometric intersection graphs.

2.3 Geometric intersection graphs

In this section we introduce the notion of intersection graphs and discuss their distance VC-dimension. Throughout the paper, the symbol $\oplus$ denotes the *Minkowski sum* of subsets of $\mathbb{R}^2$: for any $A, B \subseteq \mathbb{R}^2$, $A \oplus B := \{(a_1 + b_1, a_2 + b_2) \mid (a_1, a_2) \in A, (b_1, b_2) \in B\}$. If $a \in \mathbb{R}^2, B \subseteq \mathbb{R}^2$ then by $a \oplus B$ we mean $\{a\} \oplus B$. We also use natural scalar multiplication: for any $\lambda \in \mathbb{R}$ and $A \subseteq \mathbb{R}^2$, $\lambda \cdot A = \{(\lambda a_1, \lambda a_2) \mid (a_1, a_2) \in A\}$.

▶ **Definition 7.** *For a shape $\mathcal{F} \subseteq \mathbb{R}^2$, an* intersection graph *$I(V, \mathcal{F})$ is a simple undirected graph with vertices $V \subseteq \mathbb{R}^2$ being points on a plane, where an edge $\{v_1, v_2\}$ for $v_1 \neq v_2 \in V$ exists if and only if shapes $\mathcal{F}$ centered at v_1 and v_2 have a nonempty intersection, i.e. $(v_1 \oplus F) \cap (v_2 \oplus F) \neq \varnothing$.*

Throughout this paper we assume that $\mathcal{F}$ is closed, bounded and convex. It turns out, we can additionally assume that $\mathcal{F}$ has a center of symmetry at $(0, 0)$.

▶ **Lemma 8.** *The graph $I(V, \mathcal{F})$ is isomorphic to $I(V, \mathcal{H})$, where $H = \frac{1}{2} \cdot [\mathcal{F} \oplus (-\mathcal{F})]$.*

Proof. See the full version of the paper. ◀

Our main focus will be on the case where $\mathcal{F}$ is an s-sided polygon, however, even without this assumption, the geometric intersection graphs have bounded distance VC-dimension. The following lemma formally states that, and will be another crucial tool for our results.

▶ **Lemma 9.** *For any intersection graph $I(V, \mathcal{F})$, its distance VC-dimension is at most 4.*

Proof. A more general version of this lemma was elegantly proven in [10] using different approach, making our proof redundant. We include our proof in the full version of the paper for the sake of completeness. ◀

Please note that the definition of $I(V, \mathcal{F})$ allows the copies of the shape $\mathcal{F}$ to be translated, but not rotated. If rotation is allowed, the lemma above does not work. Moreover, even in the case of rotated triangles, there is a construction proving a conditional quadratic lower bound for the 3-Diameter problem as well as unbounded distance VC-dimension [5, 10].

2.4 ε-nets

The concept of ε-nets was introduced in [23] and applied to bounded VC-dimension hypergraphs in [11]. In this section, we recall the definitions and basic facts from these works, that we will need later.

▶ **Definition 10.** *Let $(X, \mathcal{R})$ be a hypergraph. For any $\varepsilon > 0$, a set $S \subset X$ is an* ε-net *if for every edge $R \in \mathcal{R}$, $|R| \geq \varepsilon \cdot |X| \implies R \cap S \neq \varnothing$.*

It turns out that this concept synergizes well with VC-dimension, as bounded VC-dimension implies any sufficiently large random subset to be an ε-net with high probability:

▶ **Lemma 11** ([11]). *There exists a constant α such that for any hypergraph $(X, \mathcal{R})$ of VC-dimension at most d and for any $\delta, \varepsilon > 0$, a random set $S \subseteq X$ with $|S| \geq \alpha \cdot \frac{d}{\varepsilon} \log \frac{1}{\delta\varepsilon}$ is an ε-net with probability at least $1 - \delta$.*

We will use this lemma with some modifications. First, we will employ a traditional notion of high probability, i.e. for a given c we take $\delta = |X|^{-c}$, so the probability of failure is $\frac{1}{poly(|X|)}$. Also, as in [19] we employ this lemma not for the given hypergraph $(X, \mathcal{R})$, but for $(X, \mathcal{R}^*)$, where $\mathcal{R}^* = \{(R' \triangle R'') : R', R'' \in \mathcal{R}\}$. From Lemma 5 we know that the VC-dimension of $\mathcal{R}^*$ is $\mathcal{O}(d \log d)$. Taking the definition of ε-net into account, we can reformulate Lemma 11 in the following way:

▶ **Corollary 12.** *For any positive integers c and d there exists a constant $\alpha = \alpha(c, d)$ such that for any hypergraph $(X, \mathcal{R})$ of VC-dimension at most d and for any $\varepsilon > 0$, any random set $S \subseteq X$ with $|S| \geq \alpha \cdot \frac{1}{\varepsilon} \log \frac{|X|}{\varepsilon}$ has (with probability at least $1 - |X|^{-c}$) the following property: if $R', R'' \in \mathcal{R}$ and $R' \cap S = R'' \cap S$, then $|R' \triangle R''| \leq \varepsilon \cdot |X|$.*

3 General algorithm framework

3.1 Orders on hypergraphs

The following lemma is our main tool. It provides an order of the hyperedges of any bounded distance VC-dimension hypergraph such that the difference between consecutive hyperedges is "sufficiently small". It corresponds to Theorem 1.2 in [19], but with one important difference: in our setting "sufficiently small" means that the *sum* of all differences is bounded, whereas in the previous work the bounds applied to *every one* of the differences.

▶ **Lemma 13.** *Let $(X, \mathcal{R})$ be a hypergraph with VC-dimension at most d. There exists an order $R_1, R_2, \ldots, R_{|\mathcal{R}|}$ on its hyperedges such that $\sum_{i=1}^{|\mathcal{R}|-1} |R_i \triangle R_{i+1}| = \mathcal{O}\big(|\mathcal{R}|^{1-1/d} \cdot |X|\big)$. If there is an algorithm working in time complexity $P(|X|, |\mathcal{R}|)$ which can list, for any given $x \in X$, all $R \in \mathcal{R}$ containing x, then the desired order $R_1, R_2, \ldots, R_n$ can be computed, with high probability, in time complexity $\widetilde{\mathcal{O}}\big(|\mathcal{R}|^{1+1/d} + |\mathcal{R}|^{1/d} P(|X|, |\mathcal{R}|)\big)$.*

Proof. Consider a weighted, undirected graph G with $\mathcal{R}$ as its vertex set. For any $R', R'' \in \mathcal{R}$ we define the weight of the edge (R', R'') of G as $|R' \triangle R''|$. Our immediate goal is to find a spanning tree $\mathcal{T}$ of G having total cost of edges bounded by $\mathcal{O}\big(|\mathcal{R}|^{1-1/d} \cdot |X|\big)$. If we succeed, then it is easy to obtain the desired order on $\mathcal{R}$: take an Euler tour $(R_{k_1}, R_{k_2}, \ldots, R_{k_{2|\mathcal{R}|-2}})$ of $\mathcal{T}$. We know that $\sum |R_{k_i} \triangle R_{k_{i+1}}|$ is also $\mathcal{O}\big(|\mathcal{R}|^{1-1/d} \cdot |X|\big)$, as each edge of $\mathcal{T}$ appears twice in an Euler tour. But if we delete some elements from the R_{k_i} sequence, this value can only decrease, as $|A \triangle C| \leq |A \triangle B| + |B \triangle C|$ for any finite sets A, B, C. Therefore we prune the sequence, keeping only the first instance of every element of $\mathcal{R}$, obtaining a path $(R_1, \ldots, R_{|\mathcal{R}|})$ with $\sum |R_j \triangle R_{j+1}| = \mathcal{O}\big(|\mathcal{R}|^{1-1/d} \cdot |X|\big)$. Moreover, all these operations converting $\mathcal{T}$ to the order need time complexity $\mathcal{O}(|\mathcal{R}|)$. So we can now focus on finding $\mathcal{T}$.

We will use a randomized algorithm with probability of failure at most $|X|^{-c}$, for a given integer c. Pick a random set $S \subseteq X$ with $|S| = s = |\mathcal{R}|^{1/d}$ and arrange its elements in random order $(x_1, \ldots, x_s)$. Let $\alpha = \alpha(c+1, d)$ be the constant from Corollary 12 and fix $\varepsilon = \frac{2\alpha \log |X|}{s}$. Now

$$\alpha \cdot \frac{1}{\varepsilon} \log \frac{|X|}{\varepsilon} \leq \frac{s}{2 \log |X|} \cdot (\log |X| + \log s) \leq s,$$

so S and ε satisfy the assumptions of Corollary 12. Also, let $q = \lfloor \log_2 s \rfloor$ and for every $k = 0, 1, \ldots, q$ we define $s_k = \frac{s}{2^k}$ and $S_k = \{x_1, \ldots, x_{s_k}\}$. It is easy to see that $S_q \subset S_{q-1} \subset \ldots \subset S_1 \subset S_0 = S$. Observe that every prefix S_k of S is also a random subset of X, so we can also apply Corollary 12 to it if we take $\varepsilon_k = 2^k \cdot \varepsilon$ – indeed, $\alpha \cdot \frac{1}{\varepsilon_k} \log \frac{|X|}{\varepsilon_k} \leq \alpha \cdot \frac{1}{2^k} \cdot \frac{1}{\varepsilon} \log \frac{|X|}{\varepsilon} \leq \frac{s}{2^k}$. The probability of failure for each of S_k is at most $|X|^{-(c+1)}$, so by union bound, with probability greater than $1 - |X|^{-c}$ no failure will happen.

Throughout the algorithm, we maintain the partition $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \ldots \cup \mathcal{R}_t$ into disjoint subsets – *groups*. At each step, we add some edges to $\mathcal{T}$, split some of the groups into smaller parts, and maintain the invariant that $\mathcal{T}$ is a spanning tree on the set of all groups. Initially $\mathcal{T} = \varnothing$ and the partition consists of a single group $\mathcal{R}$. Now for every $j = 1, 2, \ldots, s$ we repeat the following subroutine: every group $\mathcal{R}_i$ is split into parts $\mathcal{R}_i^0 = \{R \in \mathcal{R}_i : x_j \notin R\}$ and $\mathcal{R}_i^1 = \{R \in \mathcal{R}_i : x_j \in R\}$. If both $\mathcal{R}_i^0$ and $\mathcal{R}_i^1$ are nonempty, we pick any $R_0 \in \mathcal{R}_i^0$ and $R_1 \in \mathcal{R}_i^1$, add (R_0, R_1) to $\mathcal{T}$, and add both parts as new groups instead of $\mathcal{R}_i$. If one of the parts is empty, the other is $\mathcal{R}_i$, and we leave it as it is. After completing s steps, $\mathcal{T}$ may still not span all vertices in $\mathcal{R}$. Let us call all the edges added so far the *primary* edges, and then proceed to add new arbitrary edges to $\mathcal{T}$ until it becomes a tree. Those later edges we will call *secondary*.

Now consider the edges added to $\mathcal{T}$ between step s_k and s_{k-1} (strictly after s_k, but including s_{k-1}). For any such edge (R', R'') there must be $R' \cap S_k = R'' \cap S_k$, as (R', R'') belonged to the same group after step s_k. But as S_k is an ε_k-net, this means that $R' \triangle R'' \leq 2^k \cdot \varepsilon \cdot |X|$. On the other hand, let us count the number of groups before step s_{k-1}. For any R' and R'' belonging to different groups there must be $R' \cap S_{k-1} \neq R'' \cap S_{k-1}$, so R' and R'' induce two different subsets of S_{k-1}. But as VC-dimension of $\mathcal{R}$ does not exceed d, by Corollary 4 there can be no more than $\beta |S_{k-1}|^d = \beta(\frac{s}{2^{k-1}})^d$ different subsets of S_{k-1} induced by $\mathcal{R}$, for come constant β. This proves that before step s_{k-1} there are at most $\beta(\frac{s}{2^{k-1}})^d$ edges added to $\mathcal{T}$. The cost of edges added between steps s_k and s_{k-1} can be bounded by $\beta(\frac{s}{2^{k-1}})^d \cdot 2^k \varepsilon |X|$, and the cost of all primary edges is at most

$$\sum_{k=1}^{q} \beta \left(\frac{s}{2^{k-1}}\right)^d \cdot 2^k \varepsilon |X| = \mathcal{O}\big(q \cdot s^d \cdot \varepsilon |X|\big) = \mathcal{O}\big(q \cdot s^{d-1} \cdot |X| \cdot \log |X|\big) = \widetilde{\mathcal{O}}\Big(|\mathcal{R}|^{1-1/d} \cdot |X|\Big).$$

For the secondary edges, the bound is even simpler: for every such edge (R', R'') we already know that R' and R'' ended up in the same group, so $R' \cap S = R'' \cap S$, which means $|R' \triangle R''| \leq \varepsilon |X|$. As there are at most $|\mathcal{R}|$ such edges, we bound their cost by

$$|\mathcal{R}| \cdot \varepsilon |X| = \mathcal{O}\left(|\mathcal{R}| \cdot \frac{\log |X|}{|\mathcal{R}|^{1/d}} \cdot |X|\right) = \widetilde{\mathcal{O}}\Big(|\mathcal{R}|^{1-1/d} \cdot |X|\Big),$$

which completes the proof. As for the complexity, a single step in the first phase (for primary edges) takes $\mathcal{O}(|\mathcal{R}| + P(|X|, |\mathcal{R}|))$ time, as it has to go, for a fixed x_j, through all $R \in \mathcal{R}$ and determine if $x_j \in R$. Adding secondary edges is $\mathcal{O}(|\mathcal{R}|)$. Therefore, the total complexity is $\widetilde{\mathcal{O}}\big(|\mathcal{R}|^{1+1/d} + |\mathcal{R}|^{1/d} P(|X|, |\mathcal{R}|)\big)$. ◀

We will apply Lemma 13 to distance hypergraphs, using its two variants:

- The first one (Corollary 14) uses k-neighbourhoods as the edges of the hypergraph, so in the resulting order the adjacent vertices have similar k-neighbourhoods. We can directly compute the next neighbourhood from the previous one. This will mainly be useful for geometric intersection graphs and other implicitly given graphs.
- The second one (Corollary 15) uses duality (reverses the role of vertices and their k-neighbourhoods), which allows us to express every k-neighbourhood as a sum of sublinear number of intervals. This will be useful for the general sparse graph case.

▶ **Corollary 14.** *Let $G = (V, E)$ be a graph with distance VC-dimension at most d and let $k \in \{1, 2, \ldots, n\}$ There is an order $v_1, \ldots, v_n$ on the vertices of G such that $\sum_{i=1}^{n-1} |N^k[v_i] \triangle N^k[v_{i+1}]| = \mathcal{O}\big(n^{2-1/d}\big)$. This order can be computed, with high probability, in time complexity $\widetilde{\mathcal{O}}\big(n^{1/d} \cdot T(G)\big)$, where $T(G)$ is the complexity of a single-source distance finding algorithm (e.g. BFS).*

Proof. See the full version of the paper. ◀

Let $G = (V, E)$ be a graph, and let $\sigma = (x_1, \ldots, x_n)$ be some order on vertex set V. For any $a, b \in \{1, 2, \ldots, n\}$, $a \leq b$ let $x[a, b]$ be some interval of vertices in this order, i.e. $x[a, b] = \{x_a, x_{a+1}, \ldots, x_b\}$. Every subset $D \subset V$ can be expressed as the sum of such intervals: $D = \bigcup_{i=1}^{s} x[a_i, b_i]$ for some positive integer s, and some $a_1, \ldots, a_s, b_1, \ldots, b_s$. There exists exactly one such representation with minimal possible s – let us call it $I_\sigma(D)$, the *canonical interval representation of D* with respect to the order σ. We will omit σ and write $I(D)$ whenever it is clear from the context.

▶ **Corollary 15.** *Let $G = (V, E)$ be a graph with distance VC-dimension at most d, let $k \in \{1, 2, \ldots, n\}$, and let $\alpha : V \to \mathbb{Z}^+$ be any assignment of positive integer weights to vertices. There exists an order $v_1, \ldots, v_n$ on the vertices of G such that, with respect to that order, $\sum_{x \in V} \alpha(x) \cdot |I(N^k[x])| = \widetilde{\mathcal{O}}\Big(n^{1-1/d} \cdot \sum_{j=1}^{n} \alpha(j)\Big)$. This order can be computed, with high probability, in $\widetilde{\mathcal{O}}\big(n^{1/d} \cdot T(G)\big)$ time complexity, where $T(G)$ is the complexity of a single-source distance finding algorithm (e.g. BFS).*

Proof. See the full version of the paper. ◀

3.2 Algorithm for general sparse graphs

We now prove our main result for general graphs of bounded distance VC-dimension.

▶ **Theorem 16.** *Let $\mathcal{G}$ be a graph class with distance VC-dimension bounded by $d \geq 2$. There exists an algorithm that decides if a graph $G \in \mathcal{G}$ has diameter at most k in $\widetilde{\mathcal{O}}\big(kmn^{1-1/d}\big)$ time with high probability.*

Our algorithm builds on the work of Ducoffe et al. [18, 19], whose algorithm iteratively computes r-neighbourhoods $N^r[v]$ for all $v \in V$ and $r \in \{0, \ldots, k\}$. Note that each set $N^r[v]$ can have $\mathcal{O}(n)$ elements, so their total size can be $\mathcal{O}\big(kn^2\big)$. This means that we cannot break the quadratic barrier if we store the vertex sets explicitly.

To alleviate this, [19] uses *spanning paths with low stabbing number*, i.e. arranges vertices in a particular order such that $\max_{x \in V} |I(N^r[x])| = \widetilde{\mathcal{O}}\big(n^{1-\varepsilon_d}\big)$ for a fixed r. The authors provide a subquadratic algorithm that finds such an order with ε_d dependent only on distance VC-dimension d. Then they use interval representations to encode and operate on the r-neighbourhoods. This yields an algorithm with running time $\widetilde{\mathcal{O}}\big(kmn^{1-\varepsilon_d}\big)$.

One of our goals is to improve the constant ε_d. It is known that there always exists a vertex order with $\varepsilon_d = 1/d$ ([11]), but there is no known algorithm that computes it in subquadratic time. Instead, we observe that we can relax requirements for the vertex orders. Namely, it is sufficient to obtain low weighted average instead of maximum over interval representations. This enables us to use the algorithm given by Corollary 15.

Ball encoding. As we assumed G to be connected and non-trivial, we know that $\deg(v) \geq 1$ for each $v \in V$. Let $v_1^r, \ldots, v_n^r$ be a vertex order such that with respect to that order the following holds:

$$\sum_{x \in V} \deg(x) \cdot |I(N^r[x])| = \widetilde{\mathcal{O}}\Big(mn^{1-1/d}\Big).$$

From Corollary 15, using vertex weights $\alpha(v) = \deg(v) \geq 1$, we know that such an order exists and can be computed (with high probability) in $\widetilde{\mathcal{O}}(n^{1/d}m)$ time complexity. Our algorithm encodes r-neighbourhoods using their canonical interval representations with respect to $v_1^r, \ldots, v_n^r$. More precisely, we compute sets of intervals $\mathcal{I}_v^r = I(N^r[v])$ for all vertices $v \in V$.

▶ **Lemma 17.** *Suppose we are given the encoding for $(r-1)$-neighbourhoods, i.e. the vertex order $v_1^{r-1}, \ldots, v_n^{r-1}$ and the representations $\mathcal{I}_v^{r-1}$ for all vertices $v \in V$. Then we can compute the encoding for r-neighbourhoods in $\widetilde{\mathcal{O}}(mn^{1-1/d})$ time with high probability.*

Proof. The algorithm proceeds as follows.

1. For each vertex $v \in V$, compute the interval representation $\mathcal{I}'_v$ of $N^r[v]$ with respect to the old vertex order $v_1^{r-1}, \ldots, v_n^{r-1}$. Note that $N^r[v] = \bigcup_{x \in N[v]} N^{r-1}[x]$. This means that we can compute $\mathcal{I}'_v$ by summing the representations $\mathcal{I}_x^{r-1}$ over neighbours $x \in N[v]$. This can be done using a standard line sweep procedure in $\widetilde{\mathcal{O}}\left(\sum_{x \in N[v]} |\mathcal{I}_x^{r-1}|\right)$. Overall this step takes time $\widetilde{\mathcal{O}}\left(\sum_{v \in V} \sum_{x \in N[v]} |\mathcal{I}_x^{r-1}|\right) = \widetilde{\mathcal{O}}\left(\sum_{x \in V} \deg(x) \cdot |\mathcal{I}_x^{r-1}|\right) = \widetilde{\mathcal{O}}(mn^{1-1/d})$. The total size of all representations $\mathcal{I}'_v$ is $\widetilde{\mathcal{O}}(mn^{1-1/d})$ as well.
2. Compute the new vertex order $v_1^r, \ldots, v_n^r$ via Corollary 15. To achieve this, we only need to provide an algorithm that lists vertices of $N^r[v]$ efficiently for a given vertex $v \in V$. This can be implemented easily in $\mathcal{O}(m)$ time using breadth-first search. It follows that the vertex order can be computed in $\widetilde{\mathcal{O}}(mn^{1/d})$ time.
3. Compute the canonical interval representations $\mathcal{I}_v^r$ with respect to the new vertex order $v_1^r, \ldots, v_n^r$. We do this by transforming the representations $\mathcal{I}'_v$ as follows.
   a. Let A_i be the set of vertices $x \in V$ such that v_i^r is the left endpoint of an interval in $\mathcal{I}_x^r$. Consider a vertex $x \in A_i$ for $i \geq 2$. We have that $v_i^r \in N^r[x]$ and $v_{i-1}^r \notin N^r[x]$. This is equivalent to $x \in N^r[v_i^r]$ and $x \notin N^r[v_{i-1}^r]$. It follows that $A_i = N^r[v_i^r] \setminus N^r[v_{i-1}^r]$ for $i \geq 2$. We can thus compute A_i from interval representations $\mathcal{I}'_{v_{i-1}^r}$ and $\mathcal{I}'_{v_i^r}$ in $\widetilde{\mathcal{O}}\left(|\mathcal{I}'_{v_{i-1}^r}| + |\mathcal{I}'_{v_i^r}| + |A_i|\right)$ time using line sweep procedure. It remains to handle the case when $i = 1$. By similar argument, we obtain that $A_1 = N^r[v_1^r]$, so it is enough to list vertices in $\mathcal{I}'_{v_1^r}$. Overall, this step takes time $\widetilde{\mathcal{O}}\left(\sum_{v \in V} |\mathcal{I}'_v| + \sum_{i=1}^n |A_i|\right) = \widetilde{\mathcal{O}}(mn^{1-1/d})$.
   b. Let B_i be the set of vertices $x \in V$ such that v_i^r is the right endpoint of an interval in $\mathcal{I}_x^r$. We can compute all these sets in time $\widetilde{\mathcal{O}}(mn^{1-1/d})$, similarly as above.
   c. Recover the interval representations $\mathcal{I}_v^r$ for all $v \in V$ from the sets $A_1, \ldots A_n$ and $B_1, \ldots, B_n$. This step takes time $\sum_{i=1}^n |A_i| + |B_i| = \widetilde{\mathcal{O}}(mn^{1-1/d})$.

The total running time is $\widetilde{\mathcal{O}}(mn^{1/d} + mn^{1-1/d})$, which becomes $\widetilde{\mathcal{O}}(mn^{1-1/d})$ for $d \geq 2$. ◀

Proof of Theorem 16. We start with an arbitrary permutation of vertices $v_1^0, \ldots, v_n^0$, and trivial interval representation $\mathcal{I}_v^0 = I(\{v\})$ for each vertex $v \in V$. Then we compute the encodings of all k-neighbourhoods inductively using Lemma 17. Finally, we check if there a vertex $v \in V$ such that $\mathcal{I}_v^k \neq I(V)$. If that is the case, the diameter is larger than k. Otherwise it is at most k. ◀

4 Diameter testing for implicit graphs

In this section, we consider the diameter problem for graphs of bounded distance VC-dimension that admit implicit representations. We propose a diameter testing algorithm that relies on the existence of a certain data structure, but is independent of the number of edges. In particular, this framework can be applied for geometric intersection graphs. In Section 5 we show an implementation for unit squares. Please refer to the full version of the paper for a generalization for arbitrary convex polygons.

We begin by introducing a necessary data structure template. The *Neighbour Set Data Structure* (*NSDS* for short) maintains a family $\mathcal{T}$ of vertex subsets of a graph G under the following operations:

- $\widetilde{S}' \leftarrow \textsc{AddNeighbours}(\widetilde{S}, v)$: Given a vertex subset $\widetilde{S} \in \mathcal{T}$ and a vertex $v \in V(G)$, add a new set $\widetilde{S}' = \widetilde{S} \cup N_G[v]$ to the family $\mathcal{T}$.
- $D \leftarrow \textsc{ListDifferences}(\widetilde{S}_1, \widetilde{S}_2)$: Given vertex subsets $\widetilde{S}_1, \widetilde{S}_2 \in \mathcal{T}$, output their symmetric difference $D = \widetilde{S}_1 \triangle \widetilde{S}_2$.

Initially, the family $\mathcal{T}$ contains only the empty vertex set $\varnothing$. Throughout the whole section, we will use a tilde to mark vertex sets registered within NSDS (e.g. $\widetilde{S}$). Such vertex sets are represented implicitly by references to the data structure, so the time complexity of some operations on them may be a lot smaller than their size.

We say that a graph class $\mathcal{G}$ admits an *efficient implementation* of Neighbour Set Data Structure if the operations can be implemented in the following time complexities:

- initialization in $\widetilde{\mathcal{O}}(n)$ time (given an implicit $\mathcal{O}(n)$-size representation of a graph $G \in \mathcal{G}$);
- $\textsc{AddNeighbours}$ in $\widetilde{\mathcal{O}}(1)$ time;
- $\textsc{ListDifferences}$ in $\widetilde{\mathcal{O}}(|D|)$ time.

The remainder of this section is devoted to proving the following theorem.

▶ **Theorem 18.** *Let $\mathcal{G}$ be a graph class with distance VC-dimension bounded by $d \geq 2$. If $\mathcal{G}$ admits an efficient implementation of Neighbour Set Data Structure, then there exists an algorithm that decides if a graph $G \in \mathcal{G}$ has diameter at most k in time $\widetilde{\mathcal{O}}\big(kn^{2-1/d}\big)$ with high probability.*

4.1 Algorithm outline

In this section we describe the high-level idea of the algorithm from Theorem 18, leaving some subprocedures and other technical details to following subsections. The algorithm iteratively computes r-neighbourhoods $N^r[v]$ for all vertices $v \in V$, but in a different way than in Section 3. In particular, a different encoding of neighbourhoods is used.

Balls encoding. Let $v_1^r, \ldots, v_n^r$ be the vertex order for the r-neighbourhoods produced by Corollary 14. Observe that to apply this corollary, we only need a single-source shortest path finding algorithm, and we show that the classical BFS algorithm can be simulated using NSDS in $\widetilde{\mathcal{O}}(n)$ time (see the full version of the paper for details). Using this vertex order, the r-balls are now delta-encoded using vertex sets $D_1^r, \ldots, D_n^r$ such that:

$$N^r[v_i^r] = D_1^r \triangle \ldots \triangle D_i^r \qquad D_i^r = \begin{cases} N^r[v_1] & \text{for } i = 1 \\ N^r[v_{i-1}] \triangle N^r[v_i] & \text{for } i \in \{2, \ldots, n\} \end{cases}$$

It immediately follows from Corollary 14 that total size of all sets $D_1^r, \ldots, D_n^r$ for a fixed r is bounded by $\widetilde{\mathcal{O}}\big(n^{2-1/d}\big)$. Note that this is essentially a transposition of the encoding used in Section 3, where each ball was represented by a set of intervals. Here, each set D_i is in fact the set of these balls which have one of their intervals ending between $i-1$ and i.

Algorithm step. Suppose we have already computed the encoding for $(r-1)$-balls, i.e. the vertex order $v_1^{r-1}, \ldots, v_n^{r-1}$ and the sets $D_1^{r-1}, \ldots, D_n^{r-1}$. To compute the encoding for r-balls, we proceed as follows.

1. Build representations $\widetilde{B}_1^r, \ldots, \widetilde{B}_n^r$ of r-balls in the NSDS. Specifically, we want $\widetilde{B}_i^r = N^r[v_i^{r-1}]$, i.e. we still use the vertex order for $(r-1)$-balls. Observe that $N^r[v] = N[N^{r-1}[v]]$. To naively compute the set $\widetilde{B}_i^r$, one could invoke AddNeighbours for each vertex $v \in N^{r-1}[v_i^{r-1}]$. Clearly, such approach would be too slow. In Subsection 4.2, we provide a divide-and-conquer algorithm that builds all the representations efficiently in $\widetilde{\mathcal{O}}\big(n + \sum_i |D_i^{r-1}|\big) = \widetilde{\mathcal{O}}\big(n^{2-1/d}\big)$ time.
2. Compute new vertex order $v_1^r, \ldots, v_n^r$ using Corollary 14 in $\widetilde{\mathcal{O}}\big(n^{1+1/d}\big)$ time.
3. Let π_r be a permutation mapping vertex indices for r-balls into vertex indices for $(r-1)$-balls, i.e. $\pi_r(i) = j$ if and only if $v_i^r = v_j^{r-1}$.
4. Compute new delta-encoding $D_1^r, \ldots, D_n^r$. By definition, $D_r^i = N^r[v_{i-1}^r] \triangle N^r[v_i^r] = \widetilde{B}^r_{\pi_r(i-1)} \triangle \widetilde{B}^r_{\pi_r(i)}$ for $i \geq 2$. This means that we can compute each set D_r^i by invoking ListDifferences$(\widetilde{B}^r_{\pi_r(i-1)}, \widetilde{B}^r_{\pi_r(i)})$ on computed representations. To compute D_1^r we can use ListDifferences$(\varnothing, \widetilde{B}^r_{\pi_r(1)})$. Since ListDifferences operation is output-sensitive, the overall complexity of this step is $\widetilde{\mathcal{O}}(n + \sum_i |D_i^r|) = \widetilde{\mathcal{O}}\big(n^{2-1/d}\big)$.

The total runtime of a single step is $\widetilde{\mathcal{O}}\big(n^{1+1/d} + n^{2-1/d}\big)$, which for $d \geq 2$ becomes $\widetilde{\mathcal{O}}\big(n^{2-1/d}\big)$.

Full algorithm. We start with an arbitrary permutation of vertices $v_1^0, \ldots, v_n^0$. Moreover, we have $D_1^0 = N^0[v_1^0] = \{v_1^0\}$ and $D_i^0 = N^0[v_{i-1}^0] \triangle N^0[v_i^0] = \{v_{i-1}^0, v_i^0\}$ for $i \geq 2$. Then we compute the encodings of all k-neighbourhoods by repeatedly applying the algorithm step. Finally, we check if $D_1^k = V$ and $D_i^k = \varnothing$ for all $i \geq 2$. If that is the case, the diameter is at most k. Otherwise, it is larger than k. We obtain the final time complexity $\widetilde{\mathcal{O}}\big(kn^{2-1/d}\big)$.

4.2 Ball expansion

We now describe an algorithm that builds representations of r-balls in NSDS given a delta-encoding of $(r-1)$-balls. Specifically, we are given vertex sets $D_1^{r-1}, \ldots, D_n^{r-1} \subseteq V$ and we need to compute representations $\widetilde{B}_1^r, \ldots, \widetilde{B}_n^r$ such that $\widetilde{B}_i^r = N[D_1^{r-1} \triangle \ldots \triangle D_i^{r-1}]$. A naive approach would be to build all representations separately. Instead we use a divide-and-conquer scheme that enables us to share common parts between the computed representations.

The recursive procedure takes as input vertex sets $D_1, \ldots, D_t \subseteq V$ and a data structure representation $\widetilde{S}$. The output of the procedure are representations $\widetilde{B}_1, \ldots, \widetilde{B}_t$ such that $\widetilde{B}_i = \widetilde{S} \cup N[D_1 \triangle \ldots \triangle D_i]$. We set $\widetilde{S} = \varnothing$ for the initial call; it is used later for recursion.

We begin with reduction of common vertices. Let $C = \bigcup_{i=2}^t D_i$ and consider a vertex $v \in D_1 \setminus C$. The vertex v appears in all vertex sets of form $D_1 \triangle \ldots \triangle D_i$. This means that $N[v] \subseteq \widetilde{B}_i$ for all $i \in \{1, \ldots, n\}$. We can thus update $\widetilde{S}$ with $N[v]$ by invoking AddNeighbours$(\widetilde{S}, v)$ and remove v from D_1 without changing the output. We do this for all vertices $v \in D_1 \setminus C$. Let $\widetilde{S}'$ be the updated representation $\widetilde{S}$ and $D_1' = D_1 \cap C$ be the reduced set D_1.

If $t = 1$ then we are done: we just return the updated $\widetilde{S}' = \widetilde{B}_1$. Otherwise, we use recursion. Let $m = \lfloor t/2 \rfloor + 1$. We split the sequence into halves $D_1, \ldots, D_{m-1}$ and $D_m, \ldots, D_t$. Computing the representations $\widetilde{B}_1, \ldots, \widetilde{B}_{m-1}$ is straightforward: we recurse with $\widetilde{S}'$ and $D_1', D_2, \ldots, D_{m-1}$. Then $\widetilde{B}_i = \widetilde{S}' \cup N[D_1' \triangle D_2 \triangle \ldots \triangle D_i] = \widetilde{S} \cup N[D_1 \triangle \ldots \triangle D_i]$.

To compute $\widetilde{B}_m, \ldots, \widetilde{B}_t$, we need to take into account the sets $D_1, \ldots, D_{m-1}$. We do this by replacing D_m with $D_m' = D_1' \triangle D_2 \triangle \ldots \triangle D_m$. We recurse with $\widetilde{S}'$ and $D_m', D_{m+1}, \ldots, D_t$. Then $\widetilde{B}_i = \widetilde{S}' \cup N[D_m' \triangle D_{m+1} \triangle \ldots \triangle D_i] = \widetilde{S}' \cup N[D_1' \triangle D_2 \triangle \ldots \triangle D_i] = \widetilde{S} \cup N[D_1 \triangle \ldots \triangle D_i]$. This completes the description of the procedure.

We provide the pseudocode as Algorithm 1.

Algorithm 1 Balls expansion procedure.

```
1: function ExpandBalls(D_1, ..., D_t)
2:     return ExpandBallsRecursively(∅, D_1, ..., D_t)
3:
4: function ExpandBallsRecursively(S̃, D_1, ..., D_t)
5:     S̃' ← S̃, C ← ⋃_{i=2}^{t} D_i
6:     for all v ∈ D_1 \ C do
7:         S̃' ← AddNeighbours(S̃', v)
8:     if t = 1 then
9:         return S̃'
10:    m ← ⌊t/2⌋ + 1
11:    D'_1 ← D_1 ∩ C
12:    D'_m ← D'_1 △ D_2 △ ... △ D_m
13:    B̃_1, ..., B̃_{m-1} ← ExpandBallsRecursively(S̃', D'_1, D_2, ..., D_{m-1})
14:    B̃_m, ..., B̃_t ← ExpandBallsRecursively(S̃', D'_m, D_{m+1}, ..., D_t)
15:    return B̃_1, ..., B̃_t
```

▶ **Lemma 19.** *The* ExpandBalls *procedure works in time* $\widetilde{\mathcal{O}}\left(t + \sum_{i=1}^{t} |D_i|\right)$.

Proof. See the full version of the paper. ◀

5 Polygon intersection graphs

Here we give a concrete application of our framework to geometric intersection graphs. We start with stating our main theorem:

▶ **Theorem 20.** *There is a Monte Carlo algorithm solving the k-*Diameter *problem for the class of intersection graphs* $I(V, \mathcal{F})$*, where* $\mathcal{F}$ *is*

a) *a unit square, in* $\widetilde{\mathcal{O}}\left(k \cdot n^{\frac{7}{4}}\right)$ *time.*

b) *a convex s-sided polygon, in* $\widetilde{\mathcal{O}}\left(k \cdot n^{\frac{7}{4}}\right)$ *time, with a constant factor dependent on s.*

By Lemma 8 we can assume that polygon $\mathcal{F}$ is centrally symmetric with center of symmetry at $(0, 0)$. Recall that all intersection graphs of convex shapes have their distance VC-dimension bounded by 4 (Lemma 9). Hence, to apply Theorem 18 and complete the proof, we just need to supply a proper Neighbour Set Data Structure. This would result in an $\widetilde{\mathcal{O}}\left(k \cdot n^{2-1/d}\right) = \widetilde{\mathcal{O}}\left(k \cdot n^{7/4}\right)$ time algorithm for the k-Diameter problem. The rest of this section gives a (very) rough sketch of such a data structure, focusing mainly on the unit-square case. For a more detailed description as well as the generalization to convex polygons, see the full version of the paper.

First, observe that the neighbourhood of a vertex (point) v in a unit-square intersection graph $I(V, \square)$ consists simply of all points inside a square of side 2 centered at v. This stays true for any graph $I(V, \mathcal{F})$:

▶ **Observation 21.** *Vertex* u *is a neighbour of vertex* v *in* $I(V, \mathcal{F})$ *if and only if* $u \in v + 2\mathcal{F}$.

Proof. See the full version of the paper. ◀

Therefore, if we scale all points in V by a factor of 2, any neighbourhood is a simple $\mathcal{F}$-shape centered at a point. For the intersection graphs, we now assume that the desired NSDS stores some family $\mathcal{T}$ of subsets of V (i.e. sets of points). We can now reformulate its operations in the following way:

- MARK($\widetilde{S}, (x, y)$): Given a set $\widetilde{S} \in \mathcal{T}$ and a point $(x, y) \in \mathbb{R}^2$, add a new set $\widetilde{S}' = \widetilde{S} \cup P$ to the family $\mathcal{T}$, where $P \subseteq V$ contains the points covered by $\mathcal{F}$ centered at the point (x, y).
- LISTDIFFERENCES($\widetilde{S}_1, \widetilde{S}_2$): Given sets $\widetilde{S}_1, \widetilde{S}_2 \in \mathcal{T}$, output their symmetric difference $D = \widetilde{S}_1 \triangle \widetilde{S}_2$.

To fulfill the assumptions of Theorem 18, the MARK operation should work in $\widetilde{\mathcal{O}}(1)$ time complexity, and LISTDIFFERENCES in $\widetilde{\mathcal{O}}(|D|)$ time complexity. We also allow initialization in $\widetilde{\mathcal{O}}(n)$ time, where $n = |V|$ is the number of points.

▶ **Lemma 22.** *The following holds true:*

a) *There exists an efficient implementation of Neighbouring Set Data Structure for the unit-square intersection graphs.*

b) *Let $s \in \mathbb{N}_+$ be a constant and $\mathcal{F}$ be a convex s-sided polygon with a center of symmetry. There exists an efficient implementation of Neighbouring Set Data Structure for the intersection graphs $I(V, \mathcal{F})$.*

To complete this section, we provide a high-level overview of the proof of Lemma 22a. For more details and the general version of NSDS, see the full version of the paper.

We start by dividing the plane into horizontal strips of height 1 and focusing only on one such strip – let V be the set of points in the strip. To store subsets of V, we use a data structure called *persistent segment tree*.

Segment trees. Let $V = \{v_1, v_2, \ldots, v_k\}$, and we can make sure during initialization that the points are sorted by their x coordinate. We can also assume that k is a power of 2, adding dummy points if needed. Let $V[i, j]$ denote the set $\{v_i, v_{i+1}, \ldots, v_j\}$ for any $1 \leq i \leq j \leq k$. A *segment tree* is a complete binary tree in which every node stores information associated with some interval of points $V[i, j]$. The root corresponds to $V[1, k] = V$ and any node associated with interval $V[i, j]$ with $i < j$ has two children corresponding to $V[i, s]$ and $[s + 1, j]$, where $s = \lfloor \frac{(i+j)}{2} \rfloor$. The leaves of the tree correspond to single-element intervals. The height of this tree is clearly $\mathcal{O}(\log k)$.

A single instance of a tree stores a particular subset $\widetilde{S} \subseteq V$ in the following way: in every node z associated with an interval $[i, j]$ we keep the subset $\widetilde{S} \cap V[i, j]$. We want, however, to minimize the stored information, and instead of the whole subset $\widetilde{S} \cap V[i, j]$ we will only remember one integer – the *hash* of this subset. Formally, with every element $v \in V$ we associate a random integer (hash) $h(v)$. For every node z the subset A_z stored in this node is replaced by $\bigoplus_{v \in A_x} h(v)$, i.e. the bitwise-XOR of its elements' hashes.

Now, we must find a way to store multiple distinct sets $\widetilde{S}$, and we will achieve that by employing *persistency*.

Persistency and the LISTDIFFERENCES operation. Suppose that our tree currently stores a set $\widetilde{S}$, and we want to create and store a new set $\widetilde{S}' = \widetilde{S} \cup \{v_i\}$ by adding a single element. This change requires modifying the subset (i.e. its hash) in the node responsible for $V[i, i]$ and then going up along the path to the root, correcting the subsets in $\log_2 k$ nodes. Instead of modifying these nodes in-place, we employ a standard path copying technique. The nodes are immutable and copied whenever they are updated, with children links adjusted accordingly. In particular, a new copy of the root node will be created, and this copy will correspond to the new set $\widetilde{S}'$. Observe that this allows us, for every set $\widetilde{S}$ ever created, to reconstruct its subset stored in every node. This enables a relatively simple implementation of LISTDIFFERENCES($\widetilde{S}$, $\widetilde{S}'$): we start in the root and compare the hashes of $\widetilde{S}$ and $\widetilde{S}'$ in all nodes we visit. For a node z associated with interval $[i, j]$ we compare hashes of $\widetilde{S} \cap V[i, j]$

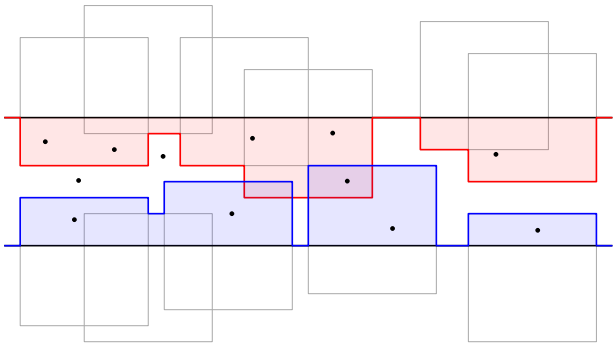

Figure 1 The top and bottom areas in a node.

and $\widetilde{S}' \cap V[i,j]$. If equal, then these subsets are equal with high probability. If not, there is at least one difference between $\widetilde{S}$ and $\widetilde{S}'$ on the interval $[i,j]$, and we recurse on both children of z. If z is a leaf, than the difference $\widetilde{S} \triangle \widetilde{S}'$ is the single element of z. An easy analysis shows that the ListDifferences works in $\mathcal{O}(|D| \cdot \log k)$ time, where D is the output set – as required. However, we have only considered simple modifications of subsets (adding one element) and our desired Mark operation needs way more.

More node data and the Mark operation. Recall that we work on a single strip of height 1. Now we want to be able to modify some subset $\widetilde{S}$ by adding to it a whole unit square centered at some point $[x,y]$ (which does not have to belong to $\widetilde{S}$, and can even lay outside of our strip, having only some part of the square inside). To achieve that, we need more information stored in every node of the tree. Recall that the points are sorted according to their x coordinate, so every node corresponds to some connected part of our strip.

A node starts with an empty set and then more and more points become marked, all points coming from some unit squares. Each square crosses either top or bottom end of the stripe, so let us call the union of the top/bottom squares the *top/bottom area*, respectively (see Figure 1, note that the top and bottom areas do not have to be disjoint). Now we keep the hashes of top and bottom areas separately, and the Mark operation hinges on the following observations:

- If the top and bottom areas are disjoint, then the hash of node's subset can be computed from the hashes of top and bottom areas;
- If a top square is added and this square covers the node's whole top area, it is easy to update the hash of the top area; the identical fact holds for bottom squares;
- If the top and bottom area together cover all the node's points, then the hash is trivial;
- Any situation not falling into above categories happens relatively rarely and adds little to the time complexity of Mark.

Due to space limitations, we defer the detailed analysis of the Mark operation to the full version of the paper.

Joining the stripes. Finally, we need to gather the information from all the stripes. Observe that any Mark only affects at most 2 stripes, so it will have the same complexity. But ListDifferences is harder – we are given sets $\widetilde{S}$ and $\widetilde{S}'$ and need to process only the stripes on which marked points are different in these two sets. To do this, we use another persistent segment tree, but with whole stripes as elements of its underlying set. For every stripe we keep the hash of its marked points, which allows us to identify the differing stripes in time complexity proportional to the number of such stripes. Then we invoke ListDifferences on them.

6 Conclusion and open problems

General Diameter problem. The algorithms presented above solve k-Diameter in subquadratic time, but not general Diameter problem. Therefore, the first important open question is:

▶ **Open Problem 1.** *What is the complexity of Diameter for geometric intersection graphs?*

The paper [19], on which we based our main algorithm, also provides an algorithm solving Diameter for K_t-minor free graphs, in $\widetilde{\mathcal{O}}\left(n^{2-\varepsilon_t}\right)$ time complexity. But unlike k-Diameter case, this algorithm does not seem to be easily translated to geometric intersection setting using our technique. We would like to briefly discuss here the obstacles we encountered, as well as some related results.

The algorithm from [19] relies on the notion of *separators*, which it uses to construct *r-divisions*. For a graph $G = (V, E)$ with $|V| = n$, a *separator* is a subset of V which has a sublinear size, and removing it would split G into connected components no larger than $\frac{2}{3}n$. An r-division is (very roughly speaking) a subset of vertices which also has sublinear size, and splits the graph into clusters of size no larger than r.

The core idea of the algorithm is to use a single-source path-finding algorithm (e.g. BFS), on vertices from the r-division of G, computing all the neighbourhoods of these vertices. This proves sufficient to determine all the other neighbourhoods.

Why does this algorithm not work in our case? It is even more surprising considering the existence of strong results involving separators in geometric intersection graphs (see [17]). However, these separators have an important difference: they are not of sublinear size by themselves, but rather can be expressed as a union of a small number of cliques. Thus we cannot run a BFS from every vertex in such a separator. It is, however, possible to use a multi-source path-finding algorithm, starting from every clique. The computed distances will differ from the exact ones only by an additive constant factor – this leads to the approximation algorithm described in [10]. To devise an exact algorithm it would be sufficient to solve the following problem: given a geometric intersection graph on the set of points V, and given some subset $A \subseteq V$ of points lying very close to each other (e.g. A fitting inside a square of small constant size δ), compute and encode the neighbourhoods of all these vertices, in subquadratic time.

Unit-disk graphs. Our data structure works with unit squares and general convex polygons, but leaves open a case of unit-disk intersection graphs:

▶ **Open Problem 2.** *Is there a subquadratic algorithm for Diameter or k-Diameter for unit-disk intersection graphs?*

This time, the main obstacle is the Neighbouring Set Data Structure: our techniques does not seem to generalize to unit disks. We would need a new way of constructing such data structures.

Lower bounds. Finally, we conjectured in Introduction that $\widetilde{\mathcal{O}}\left(n^{1-1/d} \cdot m\right)$ is a candidate for a tight complexity bound. Let us generalize this question to any lower bounds for Diameter.

▶ **Open Problem 3.** *Are there any (conditional) lower bounds for Diameter and k-Diameter for either:*

- *K_t-minor-free graphs;*
- *bounded distance VC-dimension graphs;*
- *. . . or geometric intersection graphs?*

References

1 Amir Abboud, Mina Dalirrooyfard, Ray Li, and Virginia Vassilevska Williams. On Diameter Approximation in Directed Graphs. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms (ESA 2023)*, volume 274 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:17, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2023.2.

2 Amir Abboud, Virginia Vassilevska Williams, and Joshua Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete Algorithms*, pages 377–391. SIAM, 2016.

3 Édouard Bonnet. 4 vs 7 sparse undirected unweighted diameter is seth-hard at time n 4/3. *ACM Transactions on Algorithms (TALG)*, 18(2):1–14, 2022.

4 Nicolas Bousquet and Stéphan Thomassé. Vc-dimension and erdős–pósa property. *Discrete Mathematics*, 338(12):2302–2317, 2015.

5 Karl Bringmann, Sándor Kisfaludi-Bak, Marvin Künnemann, André Nusser, and Zahra Parsaeian. Towards Sub-Quadratic Diameter Computation in Geometric Intersection Graphs. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2022.21.

6 Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Transactions on Algorithms (TALG)*, 15(2):1–38, 2018.

7 Massimo Cairo, Roberto Grossi, and Romeo Rizzi. New bounds for approximating extremal distances in undirected graphs. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 363–376. SIAM, 2016.

8 Timothy M Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In *27th International Symposium on Algorithms and Computation (ISAAC 2016)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

9 Timothy M Chan and Dimitrios Skrepetos. All-pairs shortest paths in geometric intersection graphs. In *Workshop on Algorithms and Data Structures*, pages 253–264. Springer, 2017.

10 Hsien-Chih Chang, Jie Gao, and Hung Le. Computing diameter+2 in truly subquadratic time for unit-disk graphs, 2024. arXiv:2401.12881.

11 Bernard Chazelle and Emo Welzl. Quasi-optimal range searching in spaces of finite vc-dimension. *Discrete & Computational Geometry*, 4:467–489, 1989.

12 Shiri Chechik, Daniel H Larkin, Liam Roditty, Grant Schoenebeck, Robert E Tarjan, and Virginia Vassilevska Williams. Better approximation algorithms for the graph diameter. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1041–1052. SIAM, 2014.

13 Victor Chepoi, Bertrand Estellon, and Yann Vaxes. Covering planar graphs with a fixed number of balls. *Discrete & Computational Geometry*, 37:237–244, 2007.

14 Derek G Corneil, Feodor F Dragan, Michel Habib, and Christophe Paul. Diameter determination on restricted graph families. *Discrete Applied Mathematics*, 113(2-3):143–166, 2001.

15 M. Dalirrooyfard, R. Li, and V. Williams. Hardness of approximate diameter: Now for undirected graphs. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1021–1032, Los Alamitos, CA, USA, February 2022. IEEE Computer Society. doi:10.1109/FOCS52979.2021.00102.

16 Mina Dalirrooyfard and Nicole Wein. Tight conditional lower bounds for approximating diameter in directed graphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1697–1710, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451130.

17 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for exponential-time-hypothesis–tight algorithms and lower bounds in geometric intersection graphs. *SIAM Journal on Computing*, 49(6):1291–1331, 2020. doi:10.1137/20M1320870.

18 Guillaume Ducoffe, Michel Habib, and Laurent Viennot. Diameter computation on h-minor free graphs and graphs of bounded (distance) vc-dimension. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1905–1922. SIAM, 2020.

19 Guillaume Ducoffe, Michel Habib, and Laurent Viennot. Diameter, eccentricities and distance oracle computations on h-minor free graphs and graphs of bounded (distance) vapnik–chervonenkis dimension. *SIAM Journal on Computing*, 51(5):1506–1534, 2022. doi:10.1137/20M136551X.

20 David Eisenstat and Dana Angluin. The vc dimension of k-fold union. *Information Processing Letters*, 101(5):181–184, 2007.

21 Arthur M Farley and Andrzej Proskurowski. Computation of the center and diameter of outerplanar graphs. *Discrete Applied Mathematics*, 2(3):185–191, 1980.

22 Paweł Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\widetilde{O}(n^{5/3})$ time. *SIAM Journal on Computing*, 50(2):509–554, 2021.

23 David Haussler and Emo Welzl. Epsilon-nets and simplex range queries. In *Proceedings of the second annual symposium on Computational geometry*, pages 61–71, 1986.

24 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

25 Ray Li. Improved seth-hardness of unweighted diameter. *CoRR, abs/2008.05106 v1*, 2020.

26 Ray Li. Settling seth vs. approximate sparse directed unweighted diameter (up to (nu)nseth). In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1684–1696, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451045.

27 Stephan Olariu. A simple linear-time algorithm for computing the center of an interval graph. *International Journal of Computer Mathematics*, 34(3-4):121–128, 1990.

28 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 515–524, 2013.

29 Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972.

30 Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1):247–261, 1972.

31 V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971. doi:10.1137/1116025.