From Geometry to Graphs: Geometric Data Structures and Geometric Graph Tools

Da Wei Zheng

Abstract

The thesis will be split into two parts. The first part is on geometric data structures, specifically data structures for range searching with linear and semi-algebraic ranges. These data structures are fundamental components of geometric algorithms such as for nearest neighbors, Euclidean minimum spanning tree, and computing Voronoi diagrams. The second part of the thesis is focused on developing geometric tools for special classes of graphs. There are many intriguing algorithmic questions on general graphs that have conditional lower bounds that can be bypassed in geometric and topological graphs, and so it is important to build up an algorithmic toolbox to understand the limits of these lower bounds.

Contents

Chapter 1

A journey from geometry to graphs

1.1 Introduction

This thesis reflects most of my research journey through my PhD. My early work with my advisor Timothy Chan was focused on geometric data structures. As I progressed through my PhD, I had great opportunities to work with collaborators like Yi-Jun Chang and Adam Karczmarz and use the geometric intuition I had developed to design and develop on tools for planar, minor-free, and geometric intersection graphs.

1. **Designing efficient geometric data structures [50, 52, 46, 47].** Geometric range searching and range stabbing are fundamental problems in computational geometry. My research has been on offline and online algorithms for linear and semi-algebraic ranges. Efficient data structures for these problems can also be applied to improved algorithms for problems such as higher order Voronoi diagrams [46], intersection counting of line segments and algebraic arcs in the plane [47], and Euclidean minimum spanning tree in higher dimensions [50].

2. **Developing geometric tools for graphs [57, 115, 125].** Some classes of graphs have geometric features. These graph classes include geometric intersection graphs, planar graphs, and surprisingly, even minor-free graphs. Ideas that originated from computational geometry such as cuttings, Voronoi diagrams, and VC-dimension can be abstracted and applied to these graph classes to solve fundamental problems like maximum flow, shortest path, and distance oracles. Having such tools enables us to answer questions fundamental questions of computability such as:

    - Are there ways to bypass commonly believed conjectures such as the 1-vs-2 cycle conjecture in massively parallel computing if we're given geometric information about the graph?
    - There are fine lower bounds that show for geometric intersection graphs for certain types of objects in the plane (line segments, equilateral triangles), it is unlikely to be possible to compute the diameter in truly subquadratic time in the number of objects [40]. Are there any interesting geometric intersection graphs for which we *can* compute diameter in truly subquadratic time?

Although much of my work centers on geometry-related topics in theoretical computer science, I would like to mention some of the other resesarch I've done during my PhD. I have also explored submodular optimization [118], approximation algorithms [70, 114, 172], dynamic graph data structures [118, 173], and solving optimization problems in practice [174, 135, 160] which I will not include in the thesis.

1.2 Organization of this thesis proposal

Following this, in this chapter, we give an outline of the topics and papers covered in the thesis proposal. Then, an introduction to each paper is given in the subsequent 5 chapters, mostly sourced from the original paper. Finally, in Chapter 7 we discuss some future directions of research that will be (hopefully) included in the thesis.

1.3 Geometric data structures

The prevalence of geometric data has led to a demand for fast and efficient data structures to solve problems like nearest neighbor search, point location, and range queries. These challenges arise in applications such as machine learning classification, geographic information systems, and database queries. As the volume of data grows, understanding the computational limits of what is achievable becomes increasingly important. In my research, I have been focused on improving the state of the art data structure performances for fundamental geometric problems like *range searching* and *range stabbing*.

- The **range searching problem** asks us to preprocess a set of n points, so that given a query range (e.g. a halfspace, a rectangle, a triangle, or a disk) we can efficiently report information about the set of preprocessed points that intersect with the range (e.g. give a count of the ranges or report the points in the range).
- The **range stabbing problem** is a complementary one, where we wish to preprocess a set of n ranges such that we can efficiently report information about the set of ranges stabbed by a point.

Offline range searching One of the most fundamental problem in computational geometry is halfspace range searching. The simplest version is *Hopcroft's problem* where we are given n points and n lines in the plane and we wish to know if any of the points lie on the lines. This is intimitely related to offline half-space range searching where we are given n points and n half-spaces, and the goal is to output how many points lie within each halfspace. These types of problems are important due to the relationship of the problem with other fundamental problems such as counting line segment intersections [60] and Euclidean MST [18]. This problem was intensively investigated in the '80s and '90s [75, 87, 89, 9, 60], culminating in the work of Matoušek in 1993 who gave an algorithm running in $n^{4/3}2^{O(\log^* n)}$ time [138]. In work with Timothy Chan, we gave the first improvement in almost 30 years, an algorithm that runs in $O(n^{4/3})$ time [50]. While a very small improvement, this result was very important from a theoretical perspective. For one, this closes the gap with the lower bound of $\Omega(n^{4/3})$ proven by Erickson [91]. This also gave optimal runtimes for the related problems like counting line segment intersections among n segments, Euclidean MST in dimensions three and higher, and also paved the way for later work of [47] to give an optimal algorithm for higher-order Voronoi diagrams. We go into more detail in Chapter 2 about this paper with Timothy Chan [50] that appeared in SODA 2022.

Simplex range searching and range stabbing The results of [50] imply optimal data structures for halfplane range searching and stabbing. When the ranges in question are triangles in $\mathbb{R}^2$ or more generally simplices in $\mathbb{R}^d$, range searching and range stabbing rely on *multi-level* version of the data structures for halfplanes in d dimensions, which typically loses $O(\log^d n)$ factors in the query time. In collaboration with Timothy Chan, we show in Chapter 3 that we can avoid losing these logarithmic factors [52] in our paper in SODA 2023.

Semi-algebraic range stabbing and ray shooting and intersections To handle even more complicated ranges with curved sides, we can use *semi-algebraic ranges*. Informally, a semi-algebraic range can be any range whose sides consists of a constant number of polynomial pieces, and can express objects such as disks, annuli, and polynomial slabs. In [47], together with my coauthors, we showed that the *polynomial partitioning* technique can also be used to construct efficient data structures for semi-algebraic range stabbing, ray shooting amid algebraic arcs, and counting intersections of algebraic arcs (a version of segment intersection counting when the segments are described by algebraic equations). This appears in Chapter 4 and was a collaboration with Timothy Chan and Pingan Cheng and appeared in SoCG 2024.

1.4 Geometric tools for graphs

There are fundamental graph problems such as shortest path, graph diameter, maximum flow, and constructing graph sparsifiers such as distance oracles and spanners. Often, there are lower bounds on what can be done in general graphs, but when we know that the graph comes from some specific graph class (e.g. planar graphs), we can utilize the structure of the graph class to bypass these known lower bounds. My research focuses on the following graph classes:

- **Planar graphs** are graphs that can be drawn in the plane such that no edges cross each other. They serve as useful models for practical systems such as road or rail networks, and the layouts on computer chip. Due to their relevance in real-world applications, planar graphs have been extensively studied.
- **Minor free graphs** are graphs that exclude a fixed minor H. While they can be highly expressive, they also possess many nice structural properties. Unfortunately, exploiting these structural characteristics can often be quite challenging. This graph class contains planar graphs as a special case (by Wagner's theorem, planar graphs have no K_5 or $K_{3,3}$ as a minor).
- **Geometric intersection graphs** are graphs where each vertex represent a geometric object (like a line segment or circle in the plane) and edges exist if the geometric objects intersect. When the geometric objects are unit disks, the graph is called a **unit disk graph**. These graphs can effectively model interactions of sensor networks.

The core focus of my work is leveraging my expertise in computational geometry to develop algorithmic tools and structural insights for algorithm design in these graph classes. These techniques include the use of *geometric cuttings* which are fundamental in the design of geometric data structures, and *VC dimension*, which is used to prove the existence of geometric cuttings. Surprisingly, these techniques can be generalized to work in the aforementioned graph classes, and can sometimes allow us to circumvent lower bounds that apply to general graphs and deepens our understanding of the barriers for general graphs.

Embedded planar graphs in massively parallel settings In the massively parallel model of computation (MPC), a distributed network of machines are attempting to compute information about large data sets in the fewest possible rounds of communication. This model is an abstraction of the MapReduce model, and is widely used to manage datasets too large to fit on a single machine. Assuming the widely believed 1-vs-2 cycle conjecture, solving many graph problems require at least $\Omega(\log n)$ rounds, even for simple graphs consisting of disjoint cycles. Surprisingly, in collaboration with Yi-Jun Chang, we show that this conjecture can be bypassed when the input is given as an embedded planar graph – a graph drawn with in the plane with no edge crossings and an explicit description of the edges. Many of these problems can actually be solved in $O(1)$ rounds [57] when each machine has a strongly sublinear memory.

Note that the graphs in the 1-vs-2 cycle conjecture are themselves *are* planar graphs, so our result indicate that this small amount of geometric information is enough to bypass the conjecture. To attain this result, we developed distributed algorithms for computing geometric cuttings and for graph drawing. The discussion of this work with Yi-Jun Chang of [57] that appeared in SODA 2024 is in Chapter 5.

Separators in unit disk intersection graphs Unit disk graphs are one of the simplest kind of geometric intersection graph, and results for unit disk graphs are often used as a starting point to generalize to more general geometric intersection graphs with more complicated objects. With fellow PhD students Elfarouk Harb and Zhengcheng Huang [115], we showed a structural property of unit disk graphs: every unit disk intersection graph contains a pair of shortest paths such that, when the disks along these paths and all other incident disks are removed, the graph can be partitioned into balanced components. This type of separator has been extensively utilized in the design of spanners and approximate distance oracles for planar graphs, and our results show that a similar separator also exists for unit disk graphs. This will not be discussed in this proposal for the sake of space.

Subquadratic algorithms for minor-free graphs For general sparse graphs, conditional lower bounds show that computing distance based graph quantities like graph diameter require quadratic time [152], and computing distance oracles – compact data structures for querying distances between vertices in a small (say constant) amount of time – require quadratic space [148]. For unweighted minor-free graphs, [85] show that VC dimension arguments can be used to bypass these lower bounds, yielding a subquadratic time diameter algorithm. The work of [132] extends the results to directed minor-free graphs and even achieve subquadratic space distance oracles. In a recent collaboration with Adam Karczmarz, we proved many results [125], some of the highlights being:

- a subquadratic space distance oracles for *weighted* minor-free graphs,
- a generalization of VC dimension to *psuedodimension*, which lead to an improvement in the runtime of [132] for diameter in unweighted minor-free graphs,
- and a decremental data structure for reachability in minor-free graphs with subquadratic total update time, a result not known prior for planar graphs.

We discuss the results of the paper [125] that will appear in SODA 2025 in more detail in Chapter 6.

Chapter 2

Hopcroft's problem

2.1 Introduction

In the 1980s and 1990s, many low-dimensional problems in computational geometry were discovered to have polynomial-time algorithms with fractional exponents in their time complexities. For example,

Yao [171] in 1982 described the first subquadratic algorithm for computing the Euclidean minimum spanning tree of n points in any constant dimension d, with running time near $n^{2-1/2^{d+1}}$; in the 3D case, the time bound was near $n^{9/5}$. Subsequently, the exponents were improved; in particular, in the 3D case, the best time bound was near $n^{4/3}$ [18]. For another example, Chazelle [63] in 1986 described the first subquadratic algorithm for counting the number of intersections of n line segments in 2D, with running time $O(n^{1.695})$. Eventually, the time bound was improved to near $n^{4/3}$ [111, 9, 60]. Other examples abound in the literature, too numerous to be listed here. All these problems are intimately related to range searching [19, 36], and often the exponents arise (implicitly or otherwise) from balancing the preprocessing cost and the cost of answering multiple range searching queries.

After years of research, we have now reached what are conjectured to be the optimal exponents for many fundamental geometric problems. For example, $n^{4/3}$ is likely tight for segment intersection counting: although unconditional lower bounds in general models of computation are difficult to establish, Erickson [91, 92] proved $\Omega(n^{4/3})$ lower bounds for this and other related problems in a restricted model which he called "partitioning algorithms". Erickson's lower bound applied to an even more basic problem, *Hopcroft's problem*[1]: given n points and n lines in 2D, detect (or count, or report all) point-line incidence pairs.[2] Furthermore, for some closely related range searching problems with weights, Chazelle [62, 61] proved near-$n^{4/3}$ lower bounds in the arithmetic semigroup model.

Despite the successes in this impressive body of work in computational geometry, a blemish remains: All these algorithms with fractional exponents have extra factors in their time bounds, usually of the form n^{ε} for an arbitrarily small constant $\varepsilon > 0$, or $\log^{O(1)} n$ for an undetermined number of logarithmic factors. In a few cases (see below), iterated logarithmic factors like $2^{O(\log^* n)}$ turn up, surprisingly.

For example, for Hopcroft's problem and segment intersection counting, an $O(n^{4/3+\varepsilon})$-time (randomized) algorithm was obtained by Edelsbrunner, Guibas, and Sharir [88] and Guibas, Overmars, and Sharir [111], before Agarwal [9] improved the running time to $O(n^{4/3}\log^{1.78} n)$, and then Chazelle [60] to $O(n^{4/3}\log^{1/3} n)$. For Hopcroft's problem, Matoušek [138] in 1993 managed to further reduce the bound to $n^{4/3}2^{O(\log^* n)}$.

Main new result. In this chapter, we show that Hopcroft's problem in 2D can be solved in $O(n^{4/3})$ time, without any extra factors! This improves Matoušek's result from almost 30 years ago.

Significance. Although counting incidences in 2D may not sound very useful by itself, Hopcroft's problem is significant because it is the simplest representative for a whole class of problems about nonorthogonal range searching. It reduces to offline range searching, namely, answering a batch of n range queries on n points, where the ranges are lines. Our results generalize to Hopcroft's problem in any constant dimension d (detecting/counting incidence pairs for n points and n hyperplanes in $O(n^{2d/(d+1)})$ time) and to different variants (for example, counting point-above-hyperplane pairs, or offline halfspace range counting), and indeed our ideas yield new algorithms for a long list of problems in this domain, including segment intersection counting, and Euclidean minimum spanning trees in any constant dimensions, as well as new data structures for online 2D halfplane range counting queries (see below for the statements of these results).

While shaving an iterated logarithmic factor may seem minor from the practical perspective, the value of our work is in cleaning up existing bounds for a central class of problems in computational geometry.

[1] First posed by John Hopcroft in the early 1980s.

[2] Note that the number of incidences is known to be $\Theta(n^{4/3})$ in the worst case—this is the well-known Szémeredi–Trotter Theorem [163], with lower bound construction provided by Erdös. So, $\Omega(n^{4/3})$ is a trivial lower bound on the time complexity for the "report all" version of Hopcroft's problem.

Difficulty of $\log^*$ shaving. The presence of some logarithmic factors might seem inevitable for this type of problem: for example, in data structures for 2D halfplane range counting, under standard comparison-based models, query time must be $\Omega(\log n)$ even if we allow large preprocessing time, and preprocessing time must be $\Omega(n \log n)$ even if we allow large ($o(n^{1-\varepsilon})$) query time. The known near-$n^{4/3}$ solutions to Hopcroft's problem were essentially obtained by interpolating between these two extremes. With a smart recursion, Matoušek [138] managed to lower the effect of the logarithmic factor to iterated logarithm, but recursion alone cannot get rid of the extra factors entirely, and some new ideas are needed.[3]

We actually find two different ways to achieve our improved result for Hopcroft's problem. The first approach, based on a new form of fractional cascading, works only in 2D and is randomized, but can be adapted to yield efficient data structures for online halfplane range counting queries. The second approach, based on decision trees, works in any constant dimension and is deterministic, but yields algorithms that are far less practical (and, some might say, "galactic"), although the ideas are conceptually not complicated and are theoretically quite powerful.

2.1.1 First approach via 2D fractional cascading.

The extra factors in previous algorithms to Hopcroft's problem in 2D come from the cost of multiple point location queries in arrangements of related subsets of lines. We propose an approach based on the well-known *fractional cascading* technique of Chazelle and Guibas [65, 66]. Fractional cascading allows us to search for an element in multiple lists of elements in 1D, under certain conditions, in $O(1)$ time per list instead of $O(\log n)$. The idea involves iteratively taking a fraction of the elements from one list and overlaying it with another list. Unfortunately, the technique does not extend to 2D point location in general: overlaying two planar subdivisions of linear size may create a new planar subdivision of quadratic size. In fact, Chazelle and Liu [67] formally proved lower bounds in the pointer machine model that rule out 2D fractional cascading even in very simple scenarios. (Recently, Afshani and Cheng [5] obtained some new results on 2D fractional cascading but only for *orthogonal* subdivisions.)

We show that fractional cascading, under certain conditions, is still possible for 2D arrangements of lines! The basic reason is that arrangements of lines already have quadratic size to begin with, and overlaying two such arrangements still yields an arrangement of quadratic size. One technicality is that we now need randomization when choosing a fraction of the lines. We will incorporate standard techniques by Clarkson and Shor [74] on geometric random sampling.

2.1.2 Second approach via decision trees.

Our second approach works very differently and proceeds by first bounding the algebraic decision tree complexity of the problem, i.e., we count only the cost of comparisons, and ignore the costs of all other operations that don't explicitly access the input data. Here, comparisons refer to testing the signs of constant-degree polynomial functions on a constant number of the input real numbers. It was observed [138] that for Hopcroft's problem, improved (nonuniform) decision tree bounds would automatically imply

[3] At the end of his paper [138], Matoušek wrote about the challenge in making further improvements: "... the $2^{O(\log^* n)}$ factor originates in the following manner: we are unable to solve the problem in a constant number of stages with the present method, essentialy [sic] because a nonconstant time is spent for location of every point in the cutting. Every stage contributes a constant multiplicative factor to the 'excess' in the number of subproblems. This is because we lack some mechanism to control this, similar to Chazelle's method (he can use the number of intersections of the lines as the control device, but we flip the roles of lines and points at every stage, and such a control device is missing in this situation).

"[...] All the above vague statements have a single goal—to point out although the simplex range searching problem and related questions may look completely solved, a really satisfactory solution may still await discovery."

improved (uniform) time bounds, because after a constant number of levels of Matoušek's recursion [138], the input size can be made so tiny (e.g., $o(\log\log\log n)$) that we can afford to precompute the entire decision tree for such tiny inputs. (In the algorithms literature, there are other examples of problems for which time complexity has been shown to be equivalent to decision tree complexity, such as matrix searching [131] and minimum spanning trees [146]. There are also examples of problems for which polylogarithmic speedups of algorithms were obtained by first considering the decision tree complexity, such as all-pairs shortest paths [99], 3SUM [110], and at least one other problem in computational geometry [43].)

In a seminal work, Fredman [98] showed that M values can be sorted using just $O(M)$ comparisons instead of $O(M \log M)$, under certain scenarios when the values "originate from" a smaller set of numbers. For example, the x-coordinates of the $O(n^2)$ intersections of a set of n lines can be sorted using $O(n^2)$ comparisons [161], since these $O(n^2)$ values come from $O(n)$ real numbers. (Another example from Fredman's original paper is the well-known $X+Y$ *sorting problem*, although for the decision tree complexity of that particular problem, simpler [161] and better [122] methods were later found.) We show that this type of result is not limited to the sorting problem alone, and that logarithmic-factor shavings in the decision tree setting are actually not difficult to obtain for many other problems, including point location of multiple query points in multiple subdivisions, assuming that the query points and subdivision vertices originate from $O(n)$ real numbers. The improvement for Hopcroft's problem then follows. The technique works beyond 2D and is more general than fractional cascading (which requires the subdivisions to be organized in the form of a bounded-degree tree or dag). We also discuss the framework to the decision tree complexities of other problems, although for these problems, we do not get improvements in time complexities.

2.1.3 Applications.

Some of the algorithmic consequences of our approaches include:

- An $O(n^{4/3})$-time algorithm for line segment intersection counting in 2D. The best previous bound was $O(n^{4/3}\log^{1/3} n)$ by Chazelle [60].
- An $O(n^{4/3})$-time algorithm for computing the connected components among n line segments in 2D. The best previous bound was $O(n^{4/3}\log^3 n)$ by Lopez and Thurimella [136] (a simpler alternative using biclique covers was noted by Chan [42] but still had extra logarithmic factors).
- An $O(n^{4/3})$-time randomized algorithm for bichromatic closest pair and Euclidean minimum spanning tree in 3D and in 4D. The best previous bound in 3D was $O(n^{4/3}\log^{4/3} n)$ (randomized) by Agarwal et al. [18].
- An $O(n^{4/3})$-time algorithm for the *line towering problem* in 3D (deciding whether some red line is below some blue line, given n red lines and n blue lines). The best previous bound in 3D was $O(n^{4/3+\varepsilon})$ [64].
- An $O(n^{4/3})$-time randomized algorithm for the *distance selection problem* in 2D (selecting the k-th distance among n points). The best previous randomized bound was $O(n^{4/3}\log^{2/3} n)$ [139] (see also [17, 127, 44, 167]).

The above list is not exhaustive, as more applications have followed. The above improvements may appear a bit larger than for Hopcroft's problem, but to be fair, we should mention that existing techniques can already lower many of the extra factors from these previously stated bounds (though this may have been missed in previous papers). Our new techniques are for removing the remaining $2^{O(\log^* n)}$ factors.

Our approach via 2D fractional cascading, in combination with Fredman's decision tree technique, also leads to new randomized data structures for 2D halfplane range counting, for example, achieving $P(n)=O(n^{4/3})$ expected preprocessing time and $Q(n)=O(n^{1/3})$ expected query time. This removes a $2^{O(\log^* n)}$ factor from a previous result by Chan [45]. We can get the same bounds for ray shooting among n line segments in 2D; see [45, 138, 165] for previous work. For this class of data structure problems in 2D, it is generally believed that $P(n)Q(n)^2=\Omega(n^2)$. Known data structures achieve preprocessing/query trade-offs that almost match the conjectured lower bound but with extra factors—our result is the first to eliminate all extra factors.

Chapter 3

Simplex range searching

3.1 Introduction

Simplex range searching. *Simplex range searching* is among the most fundamental and central problems in computational geometry [13, 19, 36]. Its importance cannot be overstated: countless geometric algorithms make use of simplex range searching data structures as subroutines. Given a set of n points in a constant dimension d, the goal is to build data structures so that we can quickly find the points inside a query simplex q. Several versions exist: in *range counting*, we want the number of points inside q; in *range reporting*, we want to report all the points inside q, in time proportional to the number k of output points; in *group or semigroup range query* (which generalizes range counting), we want the sum of the weights of the points inside q, assuming that each input point is given a weight from a group or semigroup.[1] Simplex ranges are fundamental because any polyhedral region can be decomposed into simplices.

After years of research, the complexity of simplex range searching is now well-understood, if we do not care about logarithmic factors. Data structures with $O(m\mathsf{poly}(\log n))$ space and $O((n/m^{1/d})\mathsf{poly}(\log n))$ query time are known (with a "$+k$" term in the query bound for the reporting version) [168, 69, 137, 138], where m is a trade-off parameter between n and n^d. These bounds are generally believed to be close to optimal.[2] The trade-off is obtained by interpolating between data structures for the two extreme cases, $m=n$ and $m=n^d$. In fact, in the linear-space regime with $m=n$, known results have even eliminated all of the extra logarithmic factors, i.e., there are data structures with $O(n)$ space and $O(n^{1-1/d})$ query time [138, 45]. However, in the large-space regime with $m=n^d$, the best query time bound known for $O(n^d\mathsf{poly}(\log n))$ space is $O(\log^{d+1} n)$, by Matoušek [138] from the 1990s. This leads to the following question:

[1] All groups and semigroups are assumed to be commutative in this paper.

[2] Notably, Chazelle [62] proved an $\Omega((n/\log n)/m^{1/d})$ lower bound on the query time for any m-space data structure in the semigroup setting; and Chazelle and Rosenberg [68] proved an $\Omega(n^{1-\varepsilon}/m^{1/d})$ lower bound on $f(n)$ for any m-space data structure with $O(f(n)+k)$ query time for simplex range reporting in the pointer machine model. It is believed that the extra factors $\log n$ and n^ε are artifacts of the proofs. (Indeed, the $\log n$ factor disappears for $d=2$ [62], and the n^ε factor has been slightly improved by Afshani [3].)

With $O(n^d \mathsf{poly}(\log n))$ space, could the query time for simplex range searching be reduced, ideally to $O(\log n)$?

Surprisingly, no progress has been reported, despite the central importance of the simplex range searching problem. Although the question may appear to be merely about shaving logarithmic factors, it is interesting for the following reasons: Matoušek's previous solution was a *multi-level* cutting tree (which we will say more about later), and there is a general feeling among researchers that the usage of multi-level data structures necessitates at least one extra logarithmic factor per level, especially when the query time is subpolynomial. Our new results will call this rule of thumb into question. Secondly, once the large-space regime is improved, potentially the entire space/query-time trade-off could be improved, by combining with the known techniques in the linear-space regime.

It is not difficult to obtain $O(\log n)$ query time if space is increased to $O(n^{d+\varepsilon})$ for an arbitrarily small constant $\varepsilon > 0$, but insisting on $O(n^d \mathsf{poly}(\log n))$ space is what makes the problem challenging. Goswami, Das, and Nandy [108] showed that for $d=2$, triangle range counting (or group range searching) queries can indeed be answered in $O(\log n)$ time with $O(n^2)$ space, but their solution was not as good for range reporting (they obtained a weaker query time bound of $O(\log^2 n+k)$) and did not extend to higher dimensions. Recently, Chan and Zheng [51] observed that for a certain range of trade-offs when m is between n and $n^{d-\varepsilon}$, the extra logarithmic factors can be eliminated for some related problems, but this makes the question for the $m=n^d$ case all the more intriguing.

We present improved data structures for simplex range query problems in any constant dimension d. With $O(n^d \mathsf{poly}(\log n))$ space, we improve the query times to $O(\log n+k)$ for simplex range reporting, and $O(\log n)$ for simplex range counting and group or semigroup queries; these query bounds are optimal. In fact, for the group or reporting version of the problem, we can even reduce space to slightly below n^d (by a small polylogarithmic factor). It is straightforward to use our results to obtain improvements for the complete space/query-time trade-off as well.

Simplex range stabbing. Another fundamental geometric data structure problem is *simplex range stabbing*: given a set of n simplices in a constant dimension d, build a data structure so that we can quickly find the simplices that are stabbed by (i.e., contain) a query point. As before, there are different versions of the problem (counting, reporting, etc.). Range stabbing may be viewed as the "inverse" of range searching, where the role of input and query objects is reversed.

The complexity of simplex range stabbing is similar to simplex range searching, if we don't care about logarithmic factors. This time, in the large-space regime, data structures with $O(n^d \mathsf{poly}(\log n))$ space and $O(\log n)$ query time follow from known techniques, but in the near-linear-space regime, known techniques give data structures with extra logarithmic factors (more precisely, $O(n\log^d n)$ space and $O(n^{1-1/d}\log^d n)$ query time [45]; see also [72] for prior work on 2D triangle stabbing with similar extra logarithmic factors). This leads to the following question:

In the near-linear space regime, could the extra logarithmic factors in the space and query time for simplex range stabbing be removed?

Again, there was a general feeling among researchers that these logarithmic factors might be necessary since the current solutions for simplex range stabbing were also multi-level data structures.

We show that *all* of the extra logarithmic factors may be eliminated! Specifically, with $O(n)$ space, we achieve $O(n^{1-1/d})$ query time for the counting or group version and $O(n^{1-1/d}+k)$ query time for the reporting version. In fact, for counting or reporting, we can even reduce the query time to slightly below $n^{1-1/d}$ (by a small polylogarithmic factor) in a computational model that allows for bit packing.

Problem	Space	Query time	Ref.
simplex reporting	n^d	$\log^{d+1} n+k$	[138]
	n^d	$\log n+k$	new
simplex counting (or group)	n^d	$\log^{d+1} n$	[138]
	n^d	$\log n$	new
simplex semigroup	n^d	$\log^{d+1} n$	[138]
	$\widetilde{O}(n^d)$	$\log n$	new
simplex stabbing reporting	$n\log^d n$	$n^{1-1/d}\log^d n+k$	[45]
	n	$n^{1-1/d}+k$	new
simplex stabbing counting (or group)	$n\log^d n$	$n^{1-1/d}\log^d n$	[45]
	n	$n^{1-1/d}$	new
segment intersection reporting in 2D	$n\log^2 n$	$\sqrt{n}\log^2 n+k$	[72]
	n	$\sqrt{n}+k$	new
segment intersection counting (or group) in 2D	$n\log^2 n$	$\sqrt{n}\log n$	[31]
	n	$\sqrt{n}$	new
segment ray shooting in 2D	$n\alpha(n)\log^2 n$	$\sqrt{n\alpha(n)}\log n$	[31]
	$n\log^2 n$	$\sqrt{n}\log n$	[72]
	$n\log n$	$\sqrt{n}\log n$	[166]
	n	$\sqrt{n}$	new

Table 3.1: Summary of new results and selected previous results. (In some of the new results, we can even get slightly below n^d space, or slightly below $n^{1-1/d}$ or $\sqrt{n}$ query time.)

Segment intersection searching and ray shooting. Lastly, we consider another related fundamental class of geometric data structure problems, this time, about line segments in 2D. Given n (possibly intersecting) line segments in 2D, we want to build data structures so that we can quickly find the input line segments intersecting a query line segment (*intersection searching*), or find the first input line segment intersected by a query ray (*ray shooting*). As before, there are different versions of intersection searching (counting, group, reporting, etc.). This class of problems has historical significance in computational geometry, having been extensively studied since the 1980s [144, 111, 11, 72, 31, 166].

The complexity of these problems are similar to triangle range searching in 2D, ignoring logarithmic factors. Recently, in SoCG'20, Wang [166] obtained improvements in the logarithmic factors in the near-linear-space regime for the ray shooting problem: his data structure achieved $O(n\log n)$ space and $O(\sqrt{n}\log n)$ query time. There was still an extra logarithmic factor in both space and time.

We obtain a new data structure that eliminates *both* logarithmic factors. With $O(n)$ space, we achieve $O(\sqrt{n})$ query time, not just for ray shooting but also for segment intersection counting or searching in the group setting, or reporting (with a "$+k$" term for reporting). In contrast, Wang's method did not extend to intersection counting. Our results even improve over previous specialized results for nonintersecting segments. In fact, for counting or reporting, we can even reduce the query time to slightly below $\sqrt{n}$ (by a small polylogarithmic factor).

Our new results are summarized in Table 5.1. (The $\widetilde{O}$ notation hides polylogarithmic factors throughout this paper.)

Chapter 4

Semialgebraic range stabbing

4.1 Introduction

The polynomial partitioning technique [113, 112] has led to a series of breakthroughs of many long-standing classic problems in computational geometry e.g., range searching [23, 140, 16], range stabbing [16], intersection searching [93, 94, 15], etc, and simplification and generalization of many existing techniques and tools [123]. Comparing to rather simple geometric objects formed by halfspaces or hyperplanes that have been studied extensively in the early days of computational geometry, polynomial partitioning enables us to attain similar results for semialgebraic sets (a set obtained by union, intersection, and complement from a set of a collection of polynomial inequalities where the number of polynomials, the number of indeterminates, and the degree of polynomials are constant). Almost all of these breakthrough results are for problems in three or higher dimensions. We complement these breakthroughs with some new results for fundamental problems involving algebraic curves in the plane.

4.1.1 Problems studied and related results

We consider the following three problems in this work.

Semialgebraic range stabbing. In this problem we are given a collection of semialgebraic sets of constant complexity in $\mathbb{R}^2$ as the input, and we want to preprocess them in a data structure so that we can quickly count or report the inputs intersected or "stabbed" by a query point (this is called a "range stabbing query", also known as a "point enclosure query"). Generalizing counting, we can also consider the *semigroup model*, where every semialgebraic set is given a value in a semigroup, and we wish to apply the semigroup operation on the values of all sets stabbed. Semialgebraic range stabbing and its "dual" problem, semialgebraic range searching, are among the most classical problems in computational geometry. The two problems are relatively well-understood for linear ranges after a decade of study by pioneers in the fields in late 80s and early 90s. We refer the readers to a survey of this topic [10]. The tools and results developed for the problems have also become textbook results [37].

However, when considering general polynomial inequalities, the problem is more difficult. Before the invention of polynomial partitioning [113, 112], there was a lack of suitable tools and few tight results were known [8]. It was only very recently [23, 140, 16], that via polynomial partitioning, efficient data structures for the two problems were found for data structures with small (near-linear) space, and data structures with very fast (polylogarithmic) query time. By interpolating the two extreme solutions,

we obtain space-time trade-offs. However, somewhat mysteriously, even if the extreme cases are almost tight, it is unknown whether the trade-off is close to optimal. For example, even for the planar annulus stabbing, there is a clear gap between the current upper bound[1] of $S(n)=O^*(n^2/Q(n)^{3/2})$ and the lower bound of $S(n)=\Omega^*(n^{3/2}/Q(n)^{3/4})$ [6] or $S(n)=\widetilde{\Omega}(n^2/Q(n)^2)$ [4], where $S(n)$ and $Q(n)$ denote space and query time respectively.

We mention that sometimes it is possible to solve certain range searching problems involving algebraic arcs more efficiently. For example, Agarwal and Sharir [25] gave improved algorithms for counting containment pairs between points and circular disks in $\mathbb{R}^2$, which can be viewed as an off-line version of either circular range searching or range stabbing. To get this improvement, they used a key technique known as "lens cutting" to cut planar curves into pseudo-segments. This allows us to use some of the classic tools developed for linear objects which are usually more efficient than their polynomial counterparts. However, to define the dual of pseudo-line or pseudo-segment arrangements, we need to know all the input and query objects in advance; that is the main reason why previous applications are restricted to offline settings. There were attempts to apply this technique to online problems [109] but to our knowledge they have not been generally successful.

Ray shooting amid algebraic arcs. We consider the problem of ray shooting where we are given a collection of algebraic arcs (of constant complexity) in $\mathbb{R}^2$ as the input, and we want to build a structure such that for any query (straight-line) ray, we can find the first arc intersecting it or assert that no such arc exists. Ray shooting is another classic problem in computational geometry with many applications in other fields such as computer graphics and robotics. Early study of ray shooting mostly centered around special cases, e.g., the input consists of line segments [12, 22], circular arcs [21], or disjoint arcs [21]. Specifically, for ray shooting queries amid line segments, it is possible to obtain a trade-off of $S(n)=O^*(n^2/Q(n)^2)$, which has been conjectured to be close to be optimal. For general algebraic curve inputs, it is possible to build an $O^*(n^2)$ space data structure with $O(\log n)$ query time in time $O^*(n^2)$ [130]. Combining the standard linear-space $O(n^{1-1/\beta})$-query time structure, we can interpolate and get a space-time trade-off curve of $S(n)=O^*(n^2/Q(n)^{\beta/(\beta-1)})$, where β is the number of parameters needed to define any polynomial in the semialgebraic sets (for bivariate polynomials of degree deg, we have $\beta\leq\binom{\deg+2}{2}-1$, but in general β is often much smaller). Very recently, Ezra and Sharir [93] showed how to answer ray shooting queries for algebraic curves of constant complexity in $\mathbb{R}^2$ with $O^*(n^{3/2})$ space and $O^*(n^{1/2})$ query time, where the exponent is independent of β. Note that this gives better $O^*(n^{3/2})$-space data structures for all $\beta>3$.

Intersection counting amid algebraic arcs. Finally, we consider intersection counting amid algebraic arcs in $\mathbb{R}^2$—more precisely, computing the sum of the number of intersections between pairs of algebraic arcs. We show new results for both online and offline versions of the problem. For the online version where we want to build data structures to count intersections with a query object, it is known that when the query object is a line segment, a structure of space-time trade-off of $S(n)=O^*(n^2/Q(n)^{3/2})$ (resp. $S(n)=O^*(n^2/Q(n)^{\beta/(\beta-1)})$) is possible for circular arcs (resp. general algebraic arcs) [130] in the plane. To the best of our knowledge, the more general problem of algebraic arc-arc intersection counting has not been studied for offline intersection counting where we are given a collection of algebraic arcs and want to count the number of intersections points. When the input consists of circular arcs, there is an $O^*(n^{3/2})$-time algorithm for the problem [24]. For more general arcs, it is unclear if any subquadratic algorithm with exponent independent of β exists.

[1] In this work, we use the notation $O^*(\cdot)$ or $\Omega^*(\cdot)$ to hide factors of n^ε where $\varepsilon>0$ is an arbitrary small constant. We use the notation $\widetilde{O}(\cdot)$ or $\widetilde{\Omega}(\cdot)$ to hide factors polylogarithmic in n.

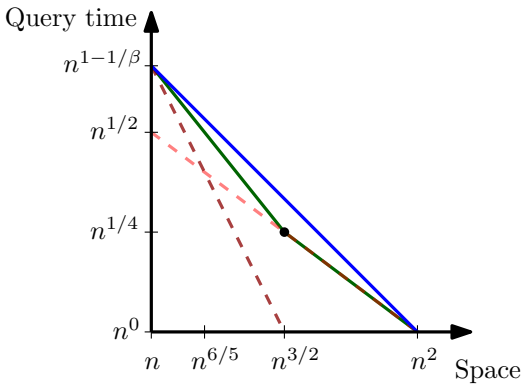

Figure 4.1: The blue line shows the prior known trade-off curve for semialgebraic range stabbing, and the green curve shows the improved trade-off curve we obtain. The dotted red lines show the lower bounds of Afshani [4] for simplex stabbing and Afshani and Cheng [6] for semialgebraic range stabbing, both of which apply.

4.1.2 New results

We present improved results for these three basic problems in 2D computational geometry.

Semialgebraic range stabbing. We give a data structure with $O^*(n^{3/2})$ preprocessing time and space and $O^*(n^{1/4})$ query time for semialgebraic range stabbing in $\mathbb{R}^2$. (This holds for counting as well as the semigroup model; for reporting, we add an $O(k)$ term to the query time where k is the output size.) Interestingly, the exponents here are independent of the number β of parameters needed to define the algebraic curves (similar phenomena have recently been seen for certain problems in 3 and higher dimensions [93, 94]). The result matches known offline results (namely, a batch of queries with $n^{5/4}$ ranges on n points take $O^*(n^{3/2})$ total time [24, 16]). By interpolating with existing results, we also automatically get an improved trade-off curve for the (online) problem. In particular, when the query time is at most $n^{1/4}$, we obtain a space-time trade-off of $S(n)Q(n)^2 = O^*(n^2)$. See Figure 4.1 for an illustration of the trade-off curve. Note that it almost matches the curve for simplex range stabbing, and is thus likely almost optimal, in this regime. Prior to our result, online data structures matching this trade-off are known only when the query time is very small (polylogarithmic), by constructing the entire $O(n^2)$-sized arrangement of the ranges.

Ray shooting amid algebraic curves. We present a data structure with $O^*(n^{3/2})$ preprocessing time and space that is able to answer ray shooting queries in time $O^*(n^{1/4})$, improving the $O^*(n^{1/2})$ query time of the previous best structure by Ezra and Sharir [93].[2] This again allows us to show a space-time trade-off that matches the one for ray shooting amid line segments when the query time satisfies $Q(n) = O^*(n^{1/4})$. Again, prior to our result, the trade-offs for the two problems only roughly match for polylogarithmic query time structures.

Intersection counting amid algebraic arcs. For the online version where we need to preprocess a collection of algebraic arcs so that we can count the intersection among them and a query algebraic arc, we give a structure with $O^*(n^{3/2})$ preprocessing time and space and $O^*(n^{1/2})$ query time. Prior

[2] To be fair, Ezra and Sharir's paper mainly focused on 3D versions of the ray shooting problem, and their 2D data structure was just one ingredient needed. However, they did consider their 2D result to be "of independent interest".

to our work, such structure is only known for when the query is a line segment instead of an algebraic arc. A straightfoward application of our online result immediately gives an $O^*(n^{3/2})$ time algorithm for the offline problem of counting the intersections of n algebraic arcs. This generalizes a known result for circular arcs [24], as well as the line segment-arc intersection detection result by Ezra and Sharir [94].

An interesting combinatorial consequence of our algorithm is that intersection graphs of algebraic arcs in $\mathbb{R}^2$ admit *biclique covers* [14, 96] of size $O^*(n^{3/2})$; it is again surprising that the exponent here is independent of β. Biclique covers have many applications to algorithmic problems about geometric intersection graphs.

Concurrent work. In an independent work, Agarwal, Ezra, and Sharir [20] showed that offline semigroup range searching with m semialgebraic ranges with β degrees of freedom and n points in $\mathbb{R}^2$ can be solved in time $O^*(m^{\frac{2\beta}{5\beta-4}}n^{\frac{5\beta-6}{5\beta-4}}+m^{2/3}n^{2/3}+m+n)$. Furthermore, they show how to compute a biclique partition of the incidence graph between the semialgebraic sets and the points. We remark that the trade-offs we get for *online* semialgebraic range stabbing directly imply both of their results.

Chapter 5

Massively parallel algorithms for embedded planar graphs

5.1 Introduction

We consider the *massively parallel computation* (MPC) model introduced by Karloff, Suri, and Vassilvitskii [126], which is a theoretical abstraction of large-scale parallel processing models such as MapReduce [81]. In comparison to the classical PRAM model of parallel computing, the MPC model allows a substantial amount of local computation in each round, making it a more realistic model for practical parallel computation. The MPC model has received much attention in recent years [27, 28, 35, 30, 76, 78, 56, 79, 82, 90, 102, 103, 120].

In the MPC model, the input is initially partitioned into machines with a memory of $\Theta(\mathcal{S})$ words. The total memory size is linear in the size of the input. For example, if the input is a graph with n vertices and m edges and $n=\Theta(m)$, then each word in the memory stores $O(\log n)$ bits and the total number of machines is $O\left(\frac{m}{\mathcal{S}}\right)$. The machines communicate with each other in synchronous rounds. In each round, each machine can send and receive $O(\mathcal{S})$ messages of $O(\log n)$ bits. After communicating with other machines, each machine can perform a local computation of $\mathsf{poly}(\mathcal{S})$ time. The main complexity measure is the number of rounds needed to solve the problem under consideration.

Our focus in this paper is on the *fully scalable* regime, where the local memory size can be an arbitrarily small polynomial, i.e., $\mathcal{S}=n^{\delta}$ for any constant $\delta>0$. Designing algorithms for this regime

is considerably more difficult than the setting where $\mathcal{S}$ is a *fixed* polynomial of n, and it has been the explicit goal of many recent papers, including [28, 29, 76, 78, 90, 106].

This paper considers planar graphs with a given embedding. We assume that each vertex is assigned a coordinate in the plane, and edges are represented by straight lines without crossings, although our approach can also accommodate edges with few crossings and more complex families of curves. This assumption is commonly observed in various applications where embeddings are readily available. For instance, real-world map data includes GPS coordinates and descriptions of roads. Often, the only way we know that a graph is planar is when we have an explicit planar embedding of the graph on the plane

In sequential models of computation, it usually does not matter if we are given an embedding or not, as there are known algorithms (e.g. [158]) to find a straight-line embedding in linear time. Surprisingly, in the MPC model, there are conditional lower bounds that indicate a separation between whether or not we have an embedding. The widely believed *1-vs-2-cycles conjecture* [33, 129, 126, 149, 156], states that distinguishing between an n-vertex cycle and two $(n/2)$-vertex cycles (both planar graphs) requires $\Omega(\log n)$ rounds in the MPC model if the local memory size $\mathcal{S}$ is at most n^{δ} for any $0<\delta<1$. Assuming this conjecture, many basic graph problems, including minimum spanning tree (MST) and counting connected components, cannot be solved in constant rounds in the MPC model in the fully scalable regime, even for planar graphs.

Such a barrier can be bypassed for some geometric problems. In particular, Andoni, Nikolov, Onak, and Yaroslavtsev [27] showed that for any constant d and ϵ, a $(1+\epsilon)$-approximate solution for the Euclidean MST problem on n points in $\mathbb{R}^d$ can be computed in $O(1)$ rounds in the MPC model in the fully scalable regime.

In contrast, for general graphs, even for the problem of computing connected components, the state-of-the-art algorithm in the MPC model in the fully scalable regime requires $O(\log D)+O(\log_{m/n}\log n)$ rounds [35], where D is the diameter of the input graph. It is unlikely that this bound can be significantly improved due to the $\Omega(\log D)$ conditional lower bound based on the 1-vs-2-cycles conjecture [35].

On the other hand, for planar graphs with a given straight-line embedding, the recent work of Holm and Tětek [119] obtained constant-round MPC algorithms for connected components, minimum spanning tree (MST), and $O(1)$-approximation of *st*-shortest path, diameter, and radius for the case where the local memory size is $\mathcal{S}=n^{2/3+\Omega(1)}$. This work showed that it is possible to bypass the 1-vs-2-cycle conjecture if we had an explicit planar embedding of a graph and had $n^{2/3+\Omega(1)}$ space per machine. Their work left one major open question:

> *Do there exist $O(1)$-round MPC algorithms for embedded planar graphs in the fully scalable regime where the local memory size is $\mathcal{S}=n^{\delta}$ for any constant $\delta>0$?*

We answer this question in the affirmative by presenting a new framework for embedded planar graphs that solves a large class of problems in the fully scalable regime. Using this framework, we give the first constant-round algorithm for connected components and MST in this regime. Furthermore, we are able to improve the approximation factor to $(1+\varepsilon)$ for *st*-shortest path, and obtain $(1+\varepsilon)$ approximations for more challenging fundamental graph problems such as single source shortest path (SSSP), all-pairs shortest path (APSP), max-flow, and min-cut. Prior work on $(1+\varepsilon)$-approximate distances, cuts, and flows required at least linear local memory $\mathcal{S}=\Omega(n)$. Our work presents the first constant-round algorithm for these fundamental problems in this more challenging memory regime.

5.1.1 Our contributions

The approach of Holm and Tětek [119] is only applicable to machines with local memory $\mathcal{S}=n^{2/3+\Omega(1)}$. This lower bound is due to the fact that the computation of the r-division has to be done in one machine, and there is a tradeoff between the local memory size and the total number of boundary vertices. The bound

	Problem	Total space	Memory per machine	Source
Embedded planar graphs	Connected Components	$O(n)$	$n^{2/3+\Omega(1)}$	[119]
		$O(n)$	n^{δ}	Theorem 5.1.1
	Minimum Spanning Tree	$O(n)$	$n^{2/3+\Omega(1)}$	[119]
		$O(n)$	n^{δ}	Theorem 5.1.2
	$O(1)$-approx. SSSP	$O(n)$	$n^{2/3+\Omega(1)}$	[119]
	$(1+\varepsilon)$-approx. SSSP	$O(n)$	n^{δ}	Theorem 5.1.5
	$(1+\varepsilon)$-approx. APSP	$O(n^2)$	n^{δ}	Theorem 5.1.7
	$(1+\varepsilon)$-approx. global min cut	$O(n)$	n^{δ}	Theorem 5.1.9
	$(1+\varepsilon)$-approx. st-max flow	$O(n)$	n^{δ}	Theorem 5.1.10
2D Euclidean MST	$(1+\varepsilon)$-approx.	$O(n)$	n^{δ}	[27]
	Exact	$O(n)$	$n^{2/3+\Omega(1)}$	[119]
	Exact	$O(n)$	n^{δ}	Corollary 5.1.3

Table 5.1: Highlights of this work in comparison to prior work. SSSP stands for single source shortest paths, and APSP stands for all pairs shortest paths. All entries take $O(1)$ rounds.

$\mathcal{S}=n^{2/3+\Omega(1)}$ is the result of balancing the two quantities. It was asked in [119, Open question 5] whether it is possible to extend their framework to the case where $\mathcal{S}=n^{2/3-\Omega(1)}$. Our first contribution is to resolve this problem by developing a new recursive framework, which allows us to extend the results in [119] to the fully scalable regime where we are allowed to set $\mathcal{S}=n^{\delta}$ for any constant $\delta>0$. Furthermore, our recursive framework can be made completely deterministic, while the algorithms of Holm and Tětek were randomized.

Theorem 5.1.1 (Connected Components)**.** *There is an algorithm that returns the number of connected components of an embedded planar graph G with n vertices in $O(1)$ rounds using $\Theta(\mathcal{S})$ space per machine and $O(n/\mathcal{S})$ machines where $\mathcal{S}=n^{\delta}$ for any constant $\delta>0$.*

Theorem 5.1.2 (Minimum Spanning Forest)**.** *There is an algorithm that returns a minimum spanning forest of an embedded planar graph G with n vertices in $O(1)$ rounds using $\Theta(\mathcal{S})$ space per machine and $O(n/\mathcal{S})$ machines where $\mathcal{S}=n^{\delta}$ for any constant $\delta>0$.*

Since Delaunay triangulations can be computed in $O(1)$ rounds [107], we obtain the first constant-round Euclidean MST algorithm for $\mathbb{R}^2$ that works in the fully scalable regime.

Corollary 5.1.3 (Euclidean MST)**.** *There is an algorithm that computes the Euclidean MST of a set P of n points in $\mathbb{R}^2$ in $O(1)$ rounds using $\Theta(\mathcal{S})$ space per machine and $O(n/\mathcal{S})$ machines where $\mathcal{S}=n^{\delta}$ for any constant $\delta>0$.*

Shortest paths. Our second contribution is to show that our framework can be combined with the recent ε-emulator of Chang, Krauthgamer, and Tan [55], and this allows us to design $O(1)$-round MPC algorithms for $(1+\varepsilon)$-approximation, for any constant $\varepsilon>0$, of *single-source shortest paths* (SSSP) and *shortest cycle* with local memory size $\mathcal{S}=n^{\delta}$, for any constant $\delta>0$. These results improve the distance computation algorithms in [119], which only achieve an approximation ratio of a fixed constant. This result resolves [119, Open question 6], which asks whether a better distance approximation is possible.

Theorem 5.1.4 ($(1+\varepsilon)$-approximate Shortest Cycle)**.** *There is an algorithm that computes the length of a $(1+\varepsilon)$-approximate shortest cycle of an embedded planar graph G with n vertices in $O(1)$ rounds using $\Theta(\mathcal{S})$ space per machine and $O(n/\mathcal{S})$ machines where $\mathcal{S}=n^\delta$ for any constant $\delta>0$.*

Theorem 5.1.5 ($(1+\varepsilon)$-approximate SSSP)**.** *There is an algorithm that computes $(1+\varepsilon)$-approximate single source shortest paths of an embedded planar graph G with n vertices in $O(1)$ rounds using $\Theta(\mathcal{S})$ space per machine and $O(n/\mathcal{S})$ machines where $\mathcal{S}=n^\delta$ for any constant $\delta>0$.*

As a corollary, we immediately obtain a constant-round MPC algorithm for $(2+\varepsilon)$-approximation for both radius and diameter in the fully scalable regime, as it is well-known that the longest shortest path distance from any given source vertex gives a 2-approximation of both radius and diameter.

Corollary 5.1.6 ($(2+\varepsilon)$-approximate diameter and radius)**.** *There is an algorithm that computes $(2+\varepsilon)$-approximate diameter and radius of an embedded planar graph G with n vertices in $O(1)$ rounds using $\Theta(\mathcal{S})$ space per machine and $O(n/\mathcal{S})$ machines where $\mathcal{S}=n^\delta$ for any constant $\delta>0$.*

Using the same ideas and $O(n^2)$ total space, we can solve the $(1+\varepsilon)$-approximate all-pairs shortest paths (APSP) problem. While this uses significantly more space than the size of the graph, $\Omega(n^2)$ total space is required to output the answer.

Theorem 5.1.7 ($(1+\varepsilon)$-approximate APSP)**.** *There is an algorithm that computes a $(1+\varepsilon)$-approximate shortest path for all pairs of vertices of an embedded planar graph G with n vertices in $O(1)$ rounds using $\Theta(\mathcal{S})$ space per machine and $O(n^2/\mathcal{S})$ machines where $\mathcal{S}=n^\delta$ for any constant $\delta>0$.*

As a corollary, this gives a method for finding $(1+\varepsilon)$-approximate diameter and radius, albeit using quadratic instead of linear total memory.

Corollary 5.1.8 ($(1+\varepsilon)$-approximate diameter and radius)**.** *There is an algorithm that computes a $(1+\varepsilon)$-approximate diameter and radius of an embedded planar graph G with n vertices in $O(1)$ rounds using $\Theta(\mathcal{S})$ space per machine and $O(n^2/\mathcal{S})$ machines where $\mathcal{S}=n^\delta$ for any constant $\delta>0$.*

Planar duals. Our third contribution is to show that a graph that contains as a minor the *dual graph* of the given embedded planar graph can be constructed in $O(1)$ rounds in the fully scalable regime. The total space needed is $O(n)$. This dual graph construction, together with our shortest cycle algorithm, implies that a $(1+\varepsilon)$-approximation of *minimum cut* and *maximum flow* can also be computed in $O(1)$ rounds in the fully scalable regime. This result resolves [119, Open question 3], which asks for an MPC algorithm for the minimum cut problem in embedded planar graphs.

Theorem 5.1.9 ($(1+\varepsilon)$-approximate global min-cut)**.** *There is an algorithm to compute the global min-cut of an embedded planar graph G with n vertices in $O(1)$ rounds using $\Theta(\mathcal{S})$ space per machine and $O(n/\mathcal{S})$ machines where $\mathcal{S}=n^\delta$ for any constant $\delta>0$.*

Theorem 5.1.10 ($(1+\varepsilon)$-approximate *st*-max-flow)**.** *There is an algorithm to compute a $(1+\varepsilon)$-approximate maximum flow between two vertices of any embedded planar graph G with n vertices in $O(1)$ rounds using $\Theta(\mathcal{S})$ space per machine and $O(n/\mathcal{S})$ machines where $\mathcal{S}=n^\delta$ for any constant $\delta>0$.*

The computations of distances, cuts, and flows are difficult problems in the MPC model in that all existing works on this topic require a local memory of at least linear size: $\mathcal{S}=\Omega(n)$ [34, 116, 104, 105]. Our work is the first one that solves these problems in the fully scalable regime.

The list of problems presented here is not exhaustive, as our recursive framework is very versatile. With this framework, we can find algorithms for variants of the problems that we discuss (e.g., computing labels for connected components, recovering shortest paths, flows, and cuts) and for problems beyond the ones presented (e.g., verifying that the embedding is planar and finding a bipartition of the graph).

Chapter 6

VC dimension in minor-free graphs

6.1 Introduction

Computing all-pairs shortest paths (APSP) in a weighted graph G, and related graph parameters concerning distances, such as the girth, diameter, radius, and Wiener index, are among the most fundamental and well-studied graph problems. It is conjectured that computing APSP requires $n^{3-o(1)}$ time [170], or more specifically $mn^{1-o(1)}$ time [134], where n and m denote the numbers of vertices and edges of G, respectively. These conditional lower bounds imply the best-known algorithms for APSP [145, 169] are near-optimal.

Surprisingly, computing most of the aformentioned single-number graph parameters is as hard as solving APSP [134, 170]. Computing the diameter, whose relationship to APSP remains unclear, is an exception here. Still, computing it requires $n^{2-o(1)}$ time unless the Strong Exponential Hypothesis fails [152]. This lower bound matches the APSP upper bound for sparse graphs up to subpolynomial factors.

Improved, truly subcubic algorithms for APSP and related problems can be obtained for *unweighted* or small-weighted dense graphs via algebraic techniques, e.g., [77, 153, 159, 175]. However, for sparse graphs, we lack non-trivial algorithms even in the unweighted case, and even if the graph is undirected. While for APSP such achievements are impossible since the output has quadratic size, for graph parameters this is not an issue. Nevertheless, truly subquadratic algorithms for the aforementioned graph parameters are ruled out in the unweighted undirected case as well by believable conjectures [2, 134, 152].

Since APSP is uninteresting in sparse graphs, one often considers a more flexible variant, the *distance oracle* problem. A distance oracle is a data structure answering distance queries for pairs of vertices in G. When designing distance oracles one would like to optimize the space and the query time, although keeping the preprocessing time small is also desirable for applications. That being said, non-trivial (i.e., with sublinear query time) exact distance oracles with subquadratic preprocessing are unlikely to exist for sparse graphs (by, e.g., [148, 80]) and whether non-trivial subquadratic-space oracles are possible for sparse graphs remains a very interesting open problem.

Graph classes. The impossibility results in sparse graphs motivate the search for algorithms exploiting structural properties of special graph classes, especially if one is interested in *exact* answers. For instance, distance oracles and computing distance-related graph parameters have been studied extensively in planar (directed) graphs, e.g., [95, 142, 143]. The breakthrough application of Voronoi diagrams [41] led to subquadratic algorithms for diameter, radius, Wiener index [41, 100], and exact distance oracles with $n^{1+o(1)}$ space and $O(\mathsf{poly}(\log n))$ query time, and $\widetilde{O}(n^{3/2})$ preprocessing time [58]. All these results assume the most general setting of real-weighted directed planar graphs.

The most powerful techniques that exploit the planarity of the graphs for distance problems, like the Monge property (used e.g. in [95]), or Voronoi diagrams, are sometimes applicable to higher genus graphs but seem completely hopeless beyond that, e.g., for apex graphs or, even more generally, K_h-minor-free graphs (which are sparse for constant h).

In this work, we study distance oracles and related graph parameters computation in K_h-minor-free *directed* graphs, for $h=O(1)$.

K_h-minor-free digraphs and VC set systems. One important structural property of planar graphs that seamlessly transfers to minor-free graphs is the existence of efficiently computable $O(\sqrt{n})$-sized balanced separators. This is enough for obtaining subquadratic solutions for some of the aforementioned problems, such as computing the girth or constructing a subquadratic-space distance oracle with sublinear query time [84]. However, balanced separators alone seem not powerful enough to break through the quadratic barrier for other graph parameters like diameter, which, in the case of planar graphs, was first enabled by Voronoi diagrams [41].

Not long after the Voronoi diagrams breakthrough for planar graphs, a new promising algorithmic approach to distance problems emerged: *VC set systems*. First applied by Li and Parter [133], analyzing the number of possible *distance patterns* via bounding their VC dimension provided an alternative way to compute the diameter of a planar graph. While the approach worked only in the *unweighted undirected* case and did not yield as efficient algorithms as the Voronoi diagrams machinery, it was robust and simple enough to be applicable e.g. in the distributed setting [133]. This robustness yields applicability of the approach to more general graph classes. Using different set systems [73], [85] showed that diameter and radius (or, more generally, the n vertex eccentricities[1]) can be computed in subquadratic time in undirected unweighted graphs with both sublinear balanced separators and constant distance VC dimension, a class that includes K_h-minor-free graphs. Recently, Le and Wulff-Nilsen [132] explored the approach further specifically for K_h-minor-free graphs, and gave improved algorithms (compared to [85]) for computing the distance-related graph parameters. In particular, they gave the first subquadratic algorithm for the Wiener index in minor-free graphs.

More importantly, [132] applied the VC set systems approach to *directed* minor-free graphs for the first time. They did so by analyzing and bounding the VC dimension of set systems used in [85] and [133] in the directed case, also giving a more general variant of the latter. Building on that, in unweighted K_h-minor-free directed graphs they obtained first subquadratic algorithms for computing vertex eccentricities. They also described a distance oracle with subquadratic space and $O(\log n)$ query time.

6.1.1 An algorithmic template

Before we discuss our contributions, it is useful to first explain a high-level overview of how VC set systems have been used to design subquadratic algorithms for minor-free graphs. (In this section, to

[1]The eccentricity of a vertex is the maximum distance from that vertex to another. Diameter is the maximum vertex eccentricity, whereas radius is the minimum one.

serve both as a warmup and to make our definitions concrete, we give a sketch of how to construct an $O(1)$-query time reachability oracle in a K_h-minor-free digraph G as a running example.[2] Note that reachability can be viewed as a finite approximation to distances.) We begin by defining the two main tools involved: VC set systems and r-divisions.

Set systems. A set system is formally a collection $\mathcal{F}$ of subsets of a ground set $\mathcal{U}$. A well-known Sauer-Shelah lemma states that if the VC dimension of $\mathcal{F}$ is bounded by d, then $|\mathcal{F}| = O\left(|\mathcal{U}|^d\right)$. Importantly, the VC dimension does not increase when *restricting* the ground set to its subset $\mathcal{U}' \subset \mathcal{U}$. As a result, the upper bound d on the VC dimension of $\mathcal{F}$ also implies $|\mathcal{F}'| = O\left(|\mathcal{U}'|^d\right)$, where $\mathcal{F}' := \{S \cap \mathcal{U}' : S \in \mathcal{F}\}$ is called the restriction of $\mathcal{F}$ to $\mathcal{U}'$.

A set system $\mathcal{S}$ used in distance-related applications typically uses V, or V annotated with some auxiliary data, as the ground set. For example, the set system $\overrightarrow{\mathcal{B}}_G \subseteq 2^V$ used in [85, 132] consists of *directed balls* in G, ranging over all possible centers in V and radii in $\mathbb{R}_{\geq 0}$. (For constructing a reachability oracle, we will use the set of balls centered at every vertex of V with infinite radius.)

r-divisions. The second crucial ingredient is the so-called r-division [97] of G. For any $r \in [1,n]$, an r-division $\mathcal{R}$ of G is a collection of $O(n/r)$ edge-induced subgraphs of G, called *pieces*, satisfying the following. First, the union of the pieces in $\mathcal{R}$ equals G. Second, each piece $P \in \mathcal{R}$ has $O(r)$ edges. Moreover, the *boundary* ∂P of a piece P, defined as the subset of vertices of P shared with some other piece in $\mathcal{R}$, has size $O(\sqrt{r})$. The boundary $\partial\mathcal{R}$ of $\mathcal{R}$ is defined as the union of the individual pieces' boundaries. An r-division of a K_h-minor-free graph (for $h = O(1)$) exists for any r and can be computed in $O(n^{1+\varepsilon})$ time, by plugging in the balanced separator algorithm of [128] into the partitioning algorithm of [97].

Combining the two. Let $\mathcal{R}$ be an r-division of G, and let $\mathcal{S}$ be a set system of VC dimension $O(1)$. Roughly speaking, we define the set of *patterns* of a piece $P \in \mathcal{R}$ to be the restriction of $\mathcal{S}$ to either $V(P)$ or ∂P, whichever better suits the application. The first basic objective when choosing the system is that the number of patterns per piece is $\mathsf{poly}(r)$, and restricting a set system with constant VC dimension to the piece's (boundary) vertices ensures this. (For our running example of a reachability oracle, we can choose the set of all balls $\overrightarrow{\mathcal{B}}_G$ which has VC dimension $h-1$. Each pattern in this set system is the set of vertices in ∂P reachable from some vertex $v \in G$. By the Sauer-Shelah lemma, the number of patterns for this set system is $O\left(|\partial P|^{h-1}\right) = O\left(r^{(h-1)/2}\right)$.)

Additionally, for any piece $P \in \mathcal{R}$ and vertex $s \notin V(P)$, one should be able to map (s,P) to one or more (but surely $o(r)$) patterns δ of the piece P, so that some desired information about the distances from s to $V(P)$ can be decoded in $o(r)$ time based exclusively on:

(1) "global" properties from s to ∂P in G like reachability or distances,

(2) "local" data associated with the pattern(s) δ that is computable in polynomial time for each such pattern based on only δ and P. (In our running example, we can let δ be the set of reachable nodes from s on the boundary ∂P. This uniquely determines the set $S(\delta,P)$ of reachable nodes from s within P.)

The information of the former kind can be computed and stored in $\widetilde{O}(n^2/\sqrt{r})$ time and space through all pairs (s,P). Computing and storing the local data takes $O(n\mathsf{poly}(r))$ time and space. Therefore, the total space required is $O(n^2/\sqrt{r} + n\mathsf{poly}(r))$. For a sufficiently small-polynomial value of r, handling both the

[2] To the best of our knowledge, this easy construction has not been described prior to our work. A description of a more general $(1+\varepsilon)$-approximate distance oracle for unweighted digraphs that is very similar can be found in the full version.

local and the global data takes $n^{2-\Theta(1)}$ time. (For reachability, the pattern's local data – the set $S(\delta,P)$ of vertices within P given that a set of vertices δ on the boundary of P are reachable – is computable in $O(r)$ time, so the oracle's construction takes $O(n^2/\sqrt{r}+nr^{(h-1)/2})$ time. Setting $r=n^{2/h}$ yields $O(n^{2-1/h})$ time.)

Given the local and global data above, another challenge lies in the subquadratic computation of patterns that the $O(n^2/r)$ pairs (s,P) map to. (For computing reachable set of vertices this is trivial to do in $O(n^2/\sqrt{r})$ time.) Finally, extracting the information from local and global data for all pairs (s,P) should also take subquadratic time. (Given a set of vertices δ on the boundary of a piece ∂P that are reachable from a vertex v, deciding whether $w\in V(P)$ is reachable from v in G amounts to testing the membership $w\in S(\delta,P)$. This takes $O(1)$ time, since $S(\delta,P)$ can be stored as a data structure that supports constant time lookups.)

6.1.2 Our contribution

In this work, we further investigate the capabilities of VC set systems (and generalizations of these set systems) for solving distance-related problems in *K_h-minor-free digraphs*. We prove results in weighted, unweighted and dynamic settings.

A weighted distance oracle. First of all, for the first time, we show the applicability of the technique to an exact distance-related problem with real edge weights (but no negative cycles). Specifically, we show:

Theorem 6.1.1. *Let G be a real-weighted K_h-minor digraph. There exists an exact distance oracle for G using $O\big(n^{2-1/(4h-1)}\big)$ space and supporting arbitrary-pair distance queries in $O(\log n)$ time.*

To the best of our knowledge, all the previous applications of VC set systems [85, 132, 133] were inherently tailored to unweighted graphs[3]. In particular, the comparable distance oracles for K_h-minor-free digraphs [132] crucially relied on the fact that G is unweighted.

One can obtain a distance oracle for weighted K_h-minor-free graphs with $O(n^2/\sqrt{r})$ space and $O(\sqrt{r})$ query time for any $r\in[1,n]$ using r-divisions alone [84], but this does not achieve polylogarithmic query times while using subquadratic space, or even a subquadratic space-query time product. Such a tradeoff (in fact an almost optimal one) is only known for planar graphs [58].

A unified system with an improved pseudodimension bound. [132] analyzed two distance VC set systems for directed minor-free graphs: the aforementioned $\overrightarrow{\mathcal{B}}_G$ (for "balls", as used in [85]) and $\overrightarrow{\mathcal{LP}}_{G,M}$ (for Li-Parter, inspired by [133]). They proved that their VC dimensions are bounded by $h-1$ and h^2, respectively; the arguments used for the two bounds differed.

[132] used patterns based on the $\overrightarrow{\mathcal{B}}_G$ system to obtain a distance oracle for unweighted graphs with space $O(n^{2-1/(2h-1)})$ and $O(\log n)$ query time[4], leaving whether such an oracle can be constructed in subquadratic time unsettled. They also showed how the patterns based on the $\overrightarrow{\mathcal{LP}}_{G,M}$ system can be used for computing all the eccentricities of a minor-free digraph in $\widetilde{O}\Big(n^{2-1/(3h^2+6)}\Big)$ time.

We propose a system $\overrightarrow{\mathcal{MB}}_{G,\Delta}$ of *multiball vectors* that generalizes both $\overrightarrow{\mathcal{B}}_G$ and $\overrightarrow{\mathcal{LP}}_{G,M}$ These objects can be thought of overlaying multiple balls centered at the same vertex with fixed differences

[3] [85] extended their results on computing diameter to integer weighted graphs, but had dependence in $\log M$ where M was the size of the largest weight. Their algorithm could only give $(1+\varepsilon)$-approximations for real-weighted graphs.

[4] [132, Section 4.3.1] miscalculated the space consumption of their directed oracle to be $O(nr^{h-3/2}+n^2/\sqrt{r})$, which led the bound $O(n^{2-1/(2h-2)})$. Their oracle actually uses $O(nr^{h-1}+n^2/\sqrt{r})$ space, which leads to the $O(n^{2-1/(2h-1)})$ bound.

in radii. Our main contribution with this system is predominantly conceptual. These multiball vectors are more natural than the Li-Parter set systems used in [132], are used in the same way for bounding the number of patterns, and are straightforward to provide dimension bounds for. In particular, we prove that the *Pollard pseudodimension* [147] (which constitutes a generalization of the VC dimension) of *multiball vectors* is at most $h-1$. This shows the upper bound on the VC dimension of $\overrightarrow{\mathcal{LP}}_{G,M}$ is $h-1$ as well, improving the h^2 bound shown by [132], and removes the discrepancy between the bound they attain in undirected vs directed graphs. It also immediately implies a tighter running time bound of $\widetilde{O}\big(n^{2-1/(3h+3)}\big)$ of the algorithm for computing eccentricities given in [132].

Faster algorithms for unweighted digraphs. While the VC dimension bounds imply limited numbers of distinct patterns per piece, the patterns are still polynomial in size, e.g., of size $\Theta(\sqrt{r})$. Processing the patterns efficiently is non-trivial and critical if one hopes for subquadratic running time in the end.

We delve into how the distance patterns can be efficiently computed and manipulated. We observe that representing their piecewise collections using the dynamic strings data structures [26, 101, 141] is very helpful to that end. This idea not only leads to an improved eccentricities algorithm for K_h-minor-free digraphs but also enables obtaining first subquadratic bounds for computing the Wiener index and constructing a distance oracle with $O(\mathsf{poly}(\log n))$ query. Formally, we show the following:

Theorem 6.1.2. *Let G be an unweighted digraph that is K_h-minor-free. Then, in $\widetilde{O}\big(n^{2-1/(3h-2)}\big)$ time one can construct an $O\big(n^{2-1/(3h-2)}\big)$-space exact distance oracle for G with $O(\log n)$ query time.*

The oracle behind Theorem 6.1.2 is the same as that given by [132] but with the parameter r of the r-division set differently. The oracle of [132] had r chosen so that the space was optimized, but it was unclear whether such an oracle (in fact, for any choice of r) could be constructed in optimal or even subquadratic time. We show that by sacrificing $o\big(n^{1/(5h)}\big)$ space and processing distance patterns efficiently, we can obtain a subquadratic oracle with near-optimal construction time. The algorithm constructing the oracle of Theorem 6.1.2 can be extended fairly easily to obtain the following.

Theorem 6.1.3. *Let G be a K_h-minor-free unweighted digraph. Then, in $\widetilde{O}\big(n^{2-1/(3h-2)}\big)$ time one can compute (1) all the vertex eccentricities of G, and (2) the Wiener index of G.*

Significantly, we show the first subquadratic algorithm for computing the Wiener index in an unweighted minor-free digraph, thus solving the problem left open by [132]. As [132] argue, their strategy to compute eccentricities could not be easily applied for the Wiener index problem.

Decremental reachability oracle and bottleneck paths. Surprisingly, we also manage to apply the VC set systems approach to a dynamic data structure problem on directed minor-free graphs. We show:

Theorem 6.1.4. *Let G be a K_h-minor-free directed graph. There exists a deterministic data structure maintaining G subject to edge deletions and supporting arbitrary-pair reachability queries in $O(1)$ time.*

The total update time of the data structure is $\widetilde{O}(n^{2-1/h})$.

Dynamic reachability and its partially dynamic variants (incremental and decremental, allowing either only edge insertions or only edge deletions, respectively) are very well-understood in general graphs from both the upper- and lower-bounds perspective, e.g. [1, 38, 39, 117, 121, 151, 155, 154, 157]. It has also been studied in planar [59, 83, 124, 162] and minor-free graphs [124]. Strikingly, so far, all the non-trivial dynamic data structures for planar and minor-free digraphs that support *arbitrary-pair* reachability queries [83, 124, 162] (let alone more general queries, e.g., about distances) have offered only polynomial query times. In

particular, the techniques used for designing near-optimal static oracles for reachability (dipath separators in [164]) and distances (Voronoi diagrams [58]) proved very challenging to be applied in the dynamic setting. Our data structure is therefore the first to achieve amortized sublinear update time per edge update and polylogarithmic query time at the same time, *even for partially dynamic planar digraphs*, where, recall, much more structure and techniques are available. Observe that for planar digraphs (that are K_5-free), the data structure has $\widetilde{O}(n^{4/5})$ amortized update time if all edges are eventually deleted. Theorem 6.1.4 constitutes the first indication that the VC set systems may also be useful in dynamic settings.

Interestingly, Theorem 6.1.4 is obtained by applying the bounded VC dimension arguments to the *bottleneck metric* ball system arising from the (unknown till the updates are finished) sequence of graph snapshots. As a by-product, our decremental reachability algorithm can be easily converted into a *bottleneck distance oracle* with $\widetilde{O}\big(n^{2-1/h}\big)$ space and preprocessing, and $O(\log n)$ query time (see the full version). The bottleneck distance oracle is in turn very useful in obtaining an $(1+\varepsilon)$-approximate distance oracle for real-weighted minor-free graphs with the same bounds. We describe that in the full version for completeness, especially since it is unclear whether our weighted exact oracle (Theorem 6.1.1) can be constructed in subquadratic time at all.

Chapter 7

Future directions

7.1 Directions in geometric data structures

My interests in computational geometry are wide-ranging, extending beyond data structures [114, 150] including designing heuristics that work well in practice on geometric optimization problems [174, 135, 160]. Moving forward, there are a few main directions of research in computational geometry that I am particularly keen on pursuing.

Designing fast algorithms using geometric data structures Recent advances in continuous optimization, combined with dynamic data structures, have lead to amazing theoretical results like the celebrated almost linear time max flow result of [71]. One of my papers does something similar using dynamic graph data structures for a linear time approximation scheme [118]. This paradigm has been relatively unexplored in computational geometry. In some recent (unpublished) work, I use a dynamic approximate nearest neighbor algorithm to design a nearly linear time approximation scheme to the many-to-many matching problem [172]. I believe this is a fruitful area of reasearch and there are many other geometric optimization problems where ideas from continuous optimization can be applied.

Semi-algebraic range searching The biggest gap left by our work [47] is a faster data structure for semi-algebraic range searching (our results only applied for range stabbing). In a sentiment shared

by the authors of [20] (personal communication), we believe it should be possible for the recent advances of polynomial partitioning to get improve results for semi-algebraic range searching. However, as we have no clear path forward, it may be a good time to investigate whether we can get improved results in more restricted settings, e.g. for a deferred data structure, or for random query order arrival models.

Lower bounds for geometric data structures via biclique covers and partitions To understand the limits of how efficient geometric data structures and algorithms, there is also an importance of lower bounds. Lower bounds for halfplane range searching and even semialgebraic range searching have been around since the 90s in the pointer machine model [68], semigroup model [61], and even a restricted model of computation involving so-called partition algorithms [91]. At the core of all of these results, involve analyzing the geometric intersection graph between ranges and points (see next section on geometric intersection graphs), and deriving lower bounds for *biclique covers* and *biclique partitions*. These objects have intimate connections to combinatorial geometry, of which there has been recently renewed attention [20, 32, 48]. In some unpublished work, I've slightly improved some of the geometric lower bounds of [7], as well as proven a biclique cover lower bound for unit disk graphs.

7.2 Directions in geometric graphs

Exact diameter in geometric intersection graphs In the work of [40], the authors rule out subquadratic time algorithms for the intersection graph of unit spheres with a conditional lower bound, leaving the question of whether a subquadratic time algorithm exists for unit disk graphs open. Two recent works have attempted to tackle this problem. The first is the work of [54] that give a subquadratic time algorithm that outputs either the diameter of a unit disk graph or the diameter+1[1], and a subquadratic space distance oracle that can also be off by an additive +1. The second is the work of [86] that give an algorithm for the intersection graph of unit squares, except the algorithm is only subquadratic time when the diameter is sufficiently small. Both algorithms leverage VC dimension of the intersection graph, albeit in different ways. Since I have worked on generalizing and extending the VC dimension techniques in [125] and am familiar with the geometric aspects of unit disk graphs [115], I think I am well situated to tackle this open problem. In some preliminary (unwritten) work, in collaboration with Timothy Chan, we have an algorithm for a truly subquadratic time algorithm for diameter in unit square graphs and unit disk graphs.

Sketching geometric intersection graphs It is a natural question to ask whether the results obtained for unit disks can be extended to general geometric intersection graphs. More generally, the question of constructing a distance oracle is about constructing small *graph sketches* – compact representations of the distances between vertices of the graph. There are other compact representations of distances like *spanners* which are subgraphs that approximately preserve distances up to some multiplicative factor. In a current ongoing collaboration with Hsien-Chih Chang and Jonathan Conroy, we have some preliminary results that show we can construct a linear sized spanner for any geometric intersection graph in the plane (also known as a string graph) that preserve distances in the graph up to a constant factor, improving upon work of [49].

[1]The paper only claims a +2 approximation, but via discussion with the authors at SoCG, there is an improvement to +1 that will be reflected in the journal version and is updated in the arxiv version[53].

References

[1] Amir Abboud and Virginia Vassilevska Williams. Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems. *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*. Pp. 434–443.

[2] Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs. *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. Pp. 377–391.

[3] Peyman Afshani. Improved Pointer Machine and I/O Lower Bounds for Simplex Range Reporting and Related Problems. *Int. J. Comput. Geom. Appl.* 23 (2013). Pp. 233–252.

[4] Peyman Afshani. Improved pointer machine and I/O lower bounds for simplex range reporting and related problems. *Internat. J. Comput. Geom. Appl.* 23 (2013). Pp. 233–251.

[5] Peyman Afshani and Pingan Cheng. 2D Generalization of Fractional Cascading on Axis-aligned Planar Subdivisions. *Proc. 61st IEEE Symposium on Foundations of Computer Science (FOCS)*. Pp. 716–727.

[6] Peyman Afshani and Pingan Cheng. Lower Bounds for Semialgebraic Range Searching and Stabbing Problems. *J. ACM*. 70 (2023). 16:1–16:26.

[7] Peyman Afshani and Pingan Cheng. Lower Bounds for Semialgebraic Range Searching and Stabbing Problems. *J. ACM*. 70 (2023). 16:1–16:26.

[8] P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.* 11 (1994). Pp. 393–418.

[9] Pankaj K. Agarwal. Parititoning Arrangements of Lines II: Applications. *Discret. Comput. Geom.* 5 (1990). Pp. 533–573.

[10] Pankaj K. Agarwal. "Range searching". In: *Handbook of Discrete and Computational Geometry*. Ed. by J. E. Goodman, J. O'Rourke, and C. Toth. CRC Press, 2016.

[11] Pankaj K. Agarwal. Ray Shooting and Other Applications of Spanning Trees with Low Stabbing Number. *SIAM J. Comput.* 21 (1992). Pp. 540–570.

[12] Pankaj K. Agarwal. Ray Shooting and Other Applications of Spanning Trees with Low Stabbing Number. *SIAM J. Comput.* 21 (1992). Pp. 540–570.

[13] Pankaj K. Agarwal. "Simplex range searching and its variants: A review". In: *Journey Through Discrete Mathematics*. Ed. by M. Loebl, J. Nešetril, and R. Thomas. Springer, 2017, pp. 1–30.

[14] Pankaj K. Agarwal, Noga Alon, Boris Aronov, and Subhash Suri. Can Visibility Graphs Be Represented Compactly?. *Discret. Comput. Geom.* 12 (1994). Pp. 347–365.

[15] Pankaj K. Agarwal, Boris Aronov, Esther Ezra, Matthew J. Katz, and Micha Sharir. "Intersection queries for flat semi-algebraic objects in three dimensions and related problems". In: *Proc. 38th International Symposium on Computational Geometry (SoCG)*. 2022, 4:1–4:14. DOI: 10.4230/lipics.socg.2022.4.

[16] Pankaj K. Agarwal, Boris Aronov, Esther Ezra, and Joshua Zahl. Efficient algorithm for generalized polynomial partitioning and its applications. *SIAM J. Comput.* 50 (2021). Pp. 760–787.

[17] Pankaj K. Agarwal, Boris Aronov, Micha Sharir, and Subhash Suri. Selecting Distances in the Plane. *Algorithmica*. 9 (1993). Pp. 495–514.

[18] Pankaj K. Agarwal, Herbert Edelsbrunner, and Otfried Schwarzkopf. Euclidean Minimum Spanning Trees and Bichromatic Closest Pairs. *Discret. Comput. Geom.* 6 (1991). Pp. 407–422.

[19] Pankaj K. Agarwal and Jeff Erickson. "Geometric range searching and its relatives". In: *Advances in Discrete and Computational Geometry*. Ed. by B. Chazelle, J. E. Goodman, and R. Pollack. AMS Press, 1999, pp. 1–56.

[20] Pankaj K. Agarwal, Esther Ezra, and Micha Sharir. Semi-Algebraic Off-Line Range Searching and Biclique Partitions in the Plane. *40th International Symposium on Computational Geometry, SoCG 2024, June 11-14, 2024, Athens, Greece*. Vol. 293. 4:1–4:15.

[21] Pankaj K. Agarwal, Marc J. van Kreveld, and Mark H. Overmars. Intersection Queries in Curved Objects. *J. Algorithms*. 15 (1993). Pp. 229–266.

[22] Pankaj K. Agarwal and Jiří Matoušek. Ray shooting and parametric search. *SIAM J. Comput.* 22 (1993). Pp. 794–806.

[23] Pankaj K. Agarwal, Jiří Matoušek, and Micha Sharir. On range searching with semialgebraic sets. II. *SIAM J. Comput.* 42 (2013). Pp. 2039–2062.

[24] Pankaj K. Agarwal, Marco Pellegrini, and Micha Sharir. Counting Circular Arc Intersections. *SIAM J. Comput.* 22 (1993). Pp. 778–793.

[25] Pankaj K. Agarwal and Micha Sharir. Pseudo-line arrangements: duality, algorithms, and applications. *SIAM J. Comput.* 34 (2005). Pp. 526–552.

[26] Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. Pattern matching in dynamic texts. *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA*. Pp. 819–828.

[27] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*. Pp. 574–583.

[28] Alexandr Andoni, Zhao Song, Clifford Stein, Zhengyu Wang, and Peilin Zhong. Parallel Graph Connectivity in Log Diameter Rounds. *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*. Pp. 674–685.

[29] Alexandr Andoni, Clifford Stein, and Peilin Zhong. Log Diameter Rounds Algorithms for 2-Vertex and 2-Edge Connectivity. *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*. Vol. 132. 14:1–14:16.

[30] Alkida Balliu, Rustam Latypov, Yannic Maus, Dennis Olivetti, and Jara Uitto. Optimal Deterministic Massively Parallel Connectivity on Forests. *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*. Pp. 2589–2631.

[31] Reuven Bar-Yehuda and Sergio Fogel. Variations on Ray Shooting. *Algorithmica*. 11 (1994). Pp. 133–145.

[32] Abdul Basit, Artem Chernikov, Sergei Starchenko, Terence Tao, and Chieu-Minh Tran. Zarankiewicz's problem for semilinear hypergraphs. *Forum of Mathematics, Sigma*. Vol. 9. e59.

[33] Paul Beame, Paraschos Koutris, and Dan Suciu. Communication Steps for Parallel Query Processing. *J. ACM*. 64 (2017). 40:1–40:58.

[34] Ruben Becker, Sebastian Forster, Andreas Karrenbauer, and Christoph Lenzen. Near-Optimal Approximate Shortest Paths and Transshipment in Distributed and Streaming Models. *SIAM J. Comput.* 50 (2021). Pp. 815–856.

[35] Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, Jakub Lacki, and Vahab S. Mirrokni. Near-Optimal Massively Parallel Graph Connectivity. *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*. Pp. 1615–1636.

[36] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. 3rd. Springer, 2008. URL: https://www.worldcat.org/oclc/227584184.

[37] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. 3rd. Springer, 2008. ISBN: 9783540779735.

[38] Aaron Bernstein, Maximilian Probst, and Christian Wulff-Nilsen. Decremental strongly-connected components and single-source reachability in near-linear time. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. Pp. 365–376.

[39] Jan van den Brand, Danupon Nanongkai, and Thatchaphol Saranurak. Dynamic Matrix Inverse: Improved Algorithms and Matching Conditional Lower Bounds. *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*. Pp. 456–480.

[40] Karl Bringmann, Sándor Kisfaludi-Bak, Marvin Künnemann, André Nusser, and Zahra Parsaeian. Towards Sub-Quadratic Diameter Computation in Geometric Intersection Graphs. *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*. Vol. 224. 21:1–21:16.

[41] Sergio Cabello. Subquadratic Algorithms for the Diameter and the Sum of Pairwise Distances in Planar Graphs. *ACM Trans. Algorithms*. 15 (2019). 21:1–21:38.

[42] Timothy M. Chan. Dynamic Subgraph Connectivity with Geometric Applications. *SIAM J. Comput.* 36 (2006). Pp. 681–694.

[43] Timothy M. Chan. Klee's Measure Problem Made Easy. *Proc. 54th IEEE Symposium on Foundations of Computer Science (FOCS)*. Pp. 410–419.

[44] Timothy M. Chan. On Enumerating and Selecting Distances. *Int. J. Comput. Geom. Appl.* 11 (2001). Pp. 291–304.

[45] Timothy M. Chan. Optimal Partition Trees. *Discret. Comput. Geom.* 47 (2012). Pp. 661–690.

[46] Timothy M. Chan, Pingan Cheng, and **Da Wei Zheng**. An Optimal Algorithm for Higher-Order Voronoi Diagrams in the Plane: The Usefulness of Nondeterminism. *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*. Pp. 4451–4463.

[47] Timothy M. Chan, Pingan Cheng, and **Da Wei Zheng**. Semialgebraic Range Stabbing, Ray Shooting, and Intersection Counting in the Plane. *40th International Symposium on Computational Geometry, SoCG 2024, June 11-14, 2024, Athens, Greece*. Vol. 293. 33:1–33:15.

[48] Timothy M. Chan and Sariel Har-Peled. On the Number of Incidences When Avoiding an Induced Biclique in Geometric Settings. *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*. Pp. 1398–1413.

[49] Timothy M. Chan and Zhengcheng Huang. Constant-Hop Spanners for More Geometric Intersection Graphs, with Even Smaller Size. *39th International Symposium on Computational Geometry, SoCG 2023, June 12-15, 2023, Dallas, Texas, USA*. Vol. 258. 23:1–23:16.

[50] Timothy M. Chan and **Da Wei Zheng**. Hopcroft's Problem, Log-Star Shaving, 2D Fractional Cascading, and Decision Trees. *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*. Pp. 190–210.

[51] Timothy M. Chan and **Da Wei Zheng**. Hopcroft's Problem, Log-Star Shaving, 2D Fractional Cascading, and Decision Trees. *Proc. 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Pp. 190–210.

[52] Timothy M. Chan and **Da Wei Zheng**. Simplex Range Searching Revisited: How to Shave Logs in Multi-Level Data Structures. *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*. Pp. 1493–1511.

[53] Hsien-Chih Chang, Jie Gao, and Hung Le. Computing Diameter+2 in Truly Subquadratic Time for Unit-Disk Graphs. *CoRR*. abs/2401.12881 (2024).

[54] Hsien-Chih Chang, Jie Gao, and Hung Le. Computing Diameter+2 in Truly-Subquadratic Time for Unit-Disk Graphs. *40th International Symposium on Computational Geometry, SoCG 2024, June 11-14, 2024, Athens, Greece*. Vol. 293. 38:1–38:14.

[55] Hsien-Chih Chang, Robert Krauthgamer, and Zihan Tan. Almost-linear ϵ-emulators for planar graphs. *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. Pp. 1311–1324.

[56] Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The Complexity of $(\Delta+1)$ Coloring in Congested Clique, Massively Parallel Computation, and Centralized Local Computation. *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*. Pp. 471–480.

[57] Yi-Jun Chang and **Da Wei Zheng**. Fully Scalable Massively Parallel Algorithms for Embedded Planar Graphs. *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*. Pp. 4410–4450.

[58] Panagiotis Charalampopoulos, Pawel Gawrychowski, Yaowei Long, Shay Mozes, Seth Pettie, Oren Weimann, and Christian Wulff-Nilsen. Almost Optimal Exact Distance Oracles for Planar Graphs. *J. ACM*. 70 (2023). 12:1–12:50.

[59] Panagiotis Charalampopoulos and Adam Karczmarz. Single-source shortest paths and strong connectivity in dynamic planar graphs. *J. Comput. Syst. Sci.* 124 (2022). Pp. 97–111.

[60] Bernard Chazelle. Cutting Hyperplanes for Divide-and-Conquer. *Discret. Comput. Geom.* 9 (1993). Pp. 145–158.

[61] Bernard Chazelle. Lower Bounds for Off-Line Range Searching. *Discret. Comput. Geom.* 17 (1997). Pp. 53–65.

[62] Bernard Chazelle. Lower bounds on the complexity of polytope range searching. *J. Amer. Math. Soc.* 2 (1989). Pp. 637–666.

[63] Bernard Chazelle. Reporting and Counting Segment Intersections. *J. Comput. Syst. Sci.* 32 (1986). Pp. 156–182.

[64] Bernard Chazelle, Herbert Edelsbrunner, Leonidas J. Guibas, Micha Sharir, and Jorge Stolfi. Lines in Space: Combinatorics and Algorithms. *Algorithmica*. 15 (1996). Pp. 428–447.

[65] Bernard Chazelle and Leonidas J. Guibas. Fractional Cascading: I. A Data Structuring Technique. *Algorithmica*. 1 (1986). Pp. 133–162.

[66] Bernard Chazelle and Leonidas J. Guibas. Fractional Cascading: II. Applications. *Algorithmica*. 1 (1986). Pp. 163–191.

[67] Bernard Chazelle and Ding Liu. Lower bounds for intersection searching and fractional cascading in higher dimension. *J. Comput. Syst. Sci.* 68 (2004). Pp. 269–284.

[68] Bernard Chazelle and Burton Rosenberg. Simplex Range Reporting on a Pointer Machine. *Comput. Geom.* 5 (1995). Pp. 237–247.

[69] Bernard Chazelle, Micha Sharir, and Emo Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*. 8 (1992). Pp. 407–429.

[70] Chandra Chekuri, Rhea Jain, Shubhang Kulkarni, **Da Wei Zheng**, and Weihao Zhu. From Directed Steiner Tree to Directed Polymatroid Steiner Tree in Planar Graphs. *32nd Annual European Symposium on Algorithms (ESA 2024)*. Vol. 308. 42:1–42:19.

[71] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Almost-Linear-Time Algorithms for Maximum Flow and Minimum-Cost Flow. *Commun. ACM*. 66 (2023). Pp. 85–92.

[72] Siu Wing Cheng and Ravi Janardan. Algorithms for ray-shooting and intersection searching. *Journal of Algorithms*. 13 (1992). Pp. 670–692.

[73] Victor Chepoi, Bertrand Estellon, and Yann Vaxès. Covering Planar Graphs with a Fixed Number of Balls. *Discret. Comput. Geom.* 37 (2007). Pp. 237–244.

[74] Kenneth L. Clarkson and Peter W. Shor. Application of Random Sampling in Computational Geometry, II. *Discret. Comput. Geom.* 4 (1989). Pp. 387–421.

[75] Richard Cole, Micha Sharir, and Chee-Keng Yap. On k-Hulls and Related Problems. *SIAM J. Comput.* 16 (1987). Pp. 61–77.

[76] Sam Coy and Artur Czumaj. Deterministic massively parallel connectivity. *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. Pp. 162–175.

[77] Marek Cygan, Harold N. Gabow, and Piotr Sankowski. Algorithmic Applications of Baur-Strassen's Theorem: Shortest Cycles, Diameter, and Matchings. *J. ACM.* 62 (2015). 28:1–28:30.

[78] Artur Czumaj, Peter Davies, and Merav Parter. Simple, Deterministic, Constant-Round Coloring in Congested Clique and MPC. *SIAM J. Comput.* 50 (2021). Pp. 1603–1626.

[79] Artur Czumaj, Jakub Lacki, Aleksander Madry, Slobodan Mitrovic, Krzysztof Onak, and Piotr Sankowski. Round Compression for Parallel Matching Algorithms. *SIAM J. Comput.* 49 (2020).

[80] Mina Dalirrooyfard, Ce Jin, Virginia Vassilevska Williams, and Nicole Wein. Approximation Algorithms and Hardness for n-Pairs Shortest Paths and All-Nodes Shortest Cycles. *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*. Pp. 290–300.

[81] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation (OSDI)*. Pp. 10–10.

[82] Laxman Dhulipala, David Durfee, Janardhan Kulkarni, Richard Peng, Saurabh Sawlani, and Xiaorui Sun. Parallel Batch-Dynamic Graphs: Algorithms and Lower Bounds. *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*. Pp. 1300–1319.

[83] Krzysztof Diks and Piotr Sankowski. Dynamic Plane Transitive Closure. *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*. Vol. 4698. Pp. 594–604.

[84] Hristo N Djidjev. Efficient algorithms for shortest path queries in planar digraphs. *International Workshop on Graph-Theoretic Concepts in Computer Science*. Pp. 151–165.

[85] 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 J. Comput.* 51 (2022). Pp. 1506–1534.

[86] Lech Duraj, Filip Konieczny, and Krzysztof Potepa. Better Diameter Algorithms for Bounded VC-Dimension Graphs and Geometric Intersection Graphs. *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*. Vol. 308. 51:1–51:18.

[87] Herbert Edelsbrunner, Leonidas J. Guibas, John Hershberger, Raimund Seidel, Micha Sharir, Jack Snoeyink, and Emo Welzl. Implicitly Representing Arrangements of Lines or Segments. *Discret. Comput. Geom.* 4 (1989). Pp. 433–466.

[88] Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. The Complexity and Construction of Many Faces in Arrangement of Lines and of Segments. *Discret. Comput. Geom.* 5 (1990). Pp. 161–196.

[89] Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. The Complexity of Many Cells in Arrangements of Planes and Related Problems. *Discret. Comput. Geom.* 5 (1990). Pp. 197–216.

[90] Alessandro Epasto, Mohammad Mahdian, Vahab S. Mirrokni, and Peilin Zhong. Massively Parallel and Dynamic Algorithms for Minimum Size Clustering. *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*. Pp. 1613–1660.

[91] Jeff Erickson. New Lower Bounds for Hopcroft's Problem. *Discret. Comput. Geom.* 16 (1996). Pp. 389–418.

[92] Jeff Erickson. On the relative complexities of some geometric problems. *Proc. 7th Canadian Conference on Computational Geometry (CCCG)*. Pp. 85–90.

[93] Esther Ezra and Micha Sharir. "Intersection searching amid tetrahedra in 4-space and efficient continuous collision detection". In: *Proc. 30th Annual European Symposium on Algorithms (ESA)*. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2022, 51:1–51:17. DOI: 10.4230/lipics.esa.2022.51.

[94] Esther Ezra and Micha Sharir. On ray shooting for triangles in 3-space and related problems. *SIAM J. Comput.* 51 (2022). Pp. 1065–1095.

[95] Jittat Fakcharoenphol and Satish Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Syst. Sci.* 72 (2006). Pp. 868–889.

[96] Tomás Feder and Rajeev Motwani. Clique Partitions, Graph Compression and Speeding-Up Algorithms. *J. Comput. Syst. Sci.* 51 (1995). Pp. 261–272.

[97] Greg N. Frederickson. Fast Algorithms for Shortest Paths in Planar Graphs, with Applications. *SIAM J. Comput.* 16 (1987). Pp. 1004–1022.

[98] Michael L. Fredman. How Good is the Information Theory Bound in Sorting?. *Theor. Comput. Sci.* 1 (1976). Pp. 355–361.

[99] Michael L. Fredman. New Bounds on the Complexity of the Shortest Path Problem. *SIAM J. Comput.* 5 (1976). Pp. 83–89.

[100] Pawel Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi Diagrams on Planar Graphs, and Computing the Diameter in Deterministic $\tilde{O}(n^{5/3})$ Time. *SIAM J. Comput.* 50 (2021). Pp. 509–554.

[101] Pawel Gawrychowski, Adam Karczmarz, Tomasz Kociumaka, Jakub Lacki, and Piotr Sankowski. Optimal Dynamic Strings. *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. Pp. 1509–1528.

[102] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*. Pp. 129–138.

[103] Mohsen Ghaffari, Fabian Kuhn, and Jara Uitto. Conditional Hardness Results for Massively Parallel Computation from Distributed Lower Bounds. *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*. Pp. 1650–1663.

[104] Mohsen Ghaffari and Krzysztof Nowicki. Congested Clique Algorithms for the Minimum Cut Problem. *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*. Pp. 357–366.

[105] Mohsen Ghaffari and Krzysztof Nowicki. Massively Parallel Algorithms for Minimum Cut. *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*. Pp. 119–128.

[106] Mohsen Ghaffari and Jara Uitto. Sparsifying Distributed Algorithms with Ramifications in Massively Parallel Computation and Centralized Local Computation. *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*. Pp. 1636–1653.

[107] Michael T. Goodrich. Randomized Fully-Scalable BSP Techniques for Multi-Searching and Convex Hull Construction (Preliminary Version). *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 5-7 January 1997, New Orleans, Louisiana, USA*. Pp. 767–776.

[108] Partha P. Goswami, Sandip Das, and Subhas C. Nandy. Triangular range counting query in 2D and its application in finding k nearest neighbors of a line segment. *Computational Geometry*. 29 (2004). Pp. 163–175.

[109] Ido Y. Grabinsky. "Deferred Data Structures for Online Disk Range Searching". MA thesis. Tel-Aviv University, 2009. URL: http://www.cs.tau.ac.il/thesis/thesis/thesis_ido.pdf.

[110] Allan Grønlund and Seth Pettie. Threesomes, Degenerates, and Love Triangles. *J. ACM.* 65 (2018). 22:1–22:25.

[111] Leonidas J. Guibas, Mark H. Overmars, and Micha Sharir. Intersecting Line Segments, Ray Shooting, and Other Applications of Geometric Partitioning Techniques. *Proc. 1st Scandinavian Workshop on Algorithm Theory (SWAT).* Pp. 64–73.

[112] Larry Guth. Polynomial partitioning for a set of varieties. *Math. Proc. Cambridge Philos. Soc.* 159 (2015). Pp. 459–469.

[113] Larry Guth and Nets Hawk Katz. On the Erdős distinct distances problem in the plane. *Ann. of Math. (2).* 181 (2015). Pp. 155–190.

[114] Sariel Har-Peled and **Da Wei Zheng**. Halving by a Thousand Cuts or Punctures. *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023.* Pp. 1385–1397.

[115] Elfarouk Harb, Zhengcheng Huang, and **Da Wei Zheng**. Shortest Path Separators in Unit Disk Graphs. *32nd Annual European Symposium on Algorithms (ESA 2024).* Vol. 308. 66:1–66:14.

[116] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A Deterministic Almost-Tight Distributed Algorithm for Approximating Single-Source Shortest Paths. *SIAM J. Comput.* 50 (2021).

[117] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture. *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015.* Pp. 21–30.

[118] Monika Henzinger, Paul Liu, Jan Vondrák, and **Da Wei Zheng**. Faster Submodular Maximization for Several Classes of Matroids. *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany.* Vol. 261. 74:1–74:18.

[119] Jacob Holm and Jakub Tětek. Massively Parallel Computation on Embedded Planar Graphs. *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023.* Pp. 4373–4408.

[120] Sungjin Im, Benjamin Moseley, and Xiaorui Sun. Efficient massively parallel methods for dynamic programming. *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017.* Pp. 798–811.

[121] Giuseppe F. Italiano. Amortized Efficiency of a Path Retrieval Data Structure. *Theor. Comput. Sci.* 48 (1986). Pp. 273–281.

[122] Daniel M. Kane, Shachar Lovett, and Shay Moran. Near-optimal Linear Decision Trees for k-SUM and Related Problems. *J. ACM.* 66 (2019). 16:1–16:18.

[123] Haim Kaplan, Jirí Matousek, and Micha Sharir. Simple Proofs of Classical Theorems in Discrete Geometry via the Guth-Katz Polynomial Partitioning Technique. *Discret. Comput. Geom.* 48 (2012). Pp. 499–517.

[124] Adam Karczmarz. Decremental Transitive Closure and Shortest Paths for Planar Digraphs and Beyond. *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018.* Pp. 73–92.

[125] Adam Karczmarz and **Da Wei Zheng**. Subquadratic algorithms in minor-free digraphs: (weighted) distance oracles, decremental reachability, and more. 2024. *Accepted to SODA 2025.*

[126] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A Model of Computation for MapReduce. *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010.* Pp. 938–948.

[127] Matthew J. Katz and Micha Sharir. An Expander-Based Approach to Geometric Optimization. *SIAM J. Comput.* 26 (1997). Pp. 1384–1408.

[128] Ken-ichi Kawarabayashi and Bruce A. Reed. A Separator Theorem in Minor-Closed Classes. *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. Pp. 153–162.

[129] Raimondas Kiveris, Silvio Lattanzi, Vahab S. Mirrokni, Vibhor Rastogi, and Sergei Vassilvitskii. Connected Components in MapReduce and Beyond. *Proceedings of the ACM Symposium on Cloud Computing, Seattle, WA, USA, November 3-5, 2014*. 18:1–18:13.

[130] Vladlen Koltun. Segment Intersection Searching Problems in General Settings. *Discret. Comput. Geom.* 30 (2003). Pp. 25–44.

[131] Lawrence L. Larmore. An Optimal Algorithm with Unknown Time Complexity for Convex Matrix Searching. *Inf. Process. Lett.* 36 (1990). Pp. 147–151.

[132] Hung Le and Christian Wulff-Nilsen. VC Set Systems in Minor-free (Di)Graphs and Applications. *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*. Pp. 5332–5360.

[133] Jason Li and Merav Parter. Planar diameter via metric compression. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. Pp. 152–163.

[134] Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight Hardness for Shortest Cycles and Paths in Sparse Graphs. *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. Pp. 1236–1252.

[135] Paul Liu, Jack Spalding-Jamieson, Brandon Zhang, and **Da Wei Zheng**. Coordinated Motion Planning Through Randomized k-Opt (CG Challenge). *37th International Symposium on Computational Geometry, SoCG 2021, June 7-11, 2021, Buffalo, NY, USA (Virtual Conference)*. Vol. 189. 64:1–64:8.

[136] Mario Alberto López and Ramakrishna Thurimella. On Computing Connected Components of Line Segments. *IEEE Trans. Computers*. 44 (1995). Pp. 597–601.

[137] Jiří Matousek. Efficient Partition Trees. *Discret. Comput. Geom.* 8 (1992). Pp. 315–334.

[138] Jirí Matousek. Range Searching with Efficient Hiearchical Cutting. *Discret. Comput. Geom.* 10 (1993). Pp. 157–182.

[139] Jiří Matoušek. Randomized Optimal Algorithm for Slope Selection. *Inf. Process. Lett.* 39 (1991). Pp. 183–187.

[140] Jiří Matoušek and Zuzana Patáková. Multilevel polynomial partitions and simplified range searching. *Discrete Comput. Geom.* 54 (2015). Pp. 22–41.

[141] Kurt Mehlhorn, R. Sundar, and Christian Uhrig. Maintaining Dynamic Sequences under Equality Tests in Polylogarithmic Time. *Algorithmica*. 17 (1997). Pp. 183–198.

[142] Shay Mozes, Kirill Nikolaev, Yahav Nussbaum, and Oren Weimann. Minimum Cut of Directed Planar Graphs in $O(n \log \log n)$ Time. *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. Pp. 477–494.

[143] Shay Mozes and Christian Sommer. Exact distance oracles for planar graphs. *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*. Pp. 209–222.

[144] Mark H. Overmars, Haijo Schipper, and Micha Sharir. Storing Line Segments in Partition Trees. *BIT*. 30 (1990). Pp. 385–403.

[145] Seth Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theor. Comput. Sci.* 312 (2004). Pp. 47–74.

[146] Seth Pettie and Vijaya Ramachandran. An optimal minimum spanning tree algorithm. *J. ACM.* 49 (2002). Pp. 16–34.

[147] David Pollard. Empirical Processes: Theory and Applications. *NSF-CBMS Regional Conference Series in Probability and Statistics.* 2 (1990). Pp. i–86.

[148] Mihai Puatracscu and Liam Roditty. Distance Oracles beyond the Thorup-Zwick Bound. *SIAM J. Comput.* 43 (2014). Pp. 300–311.

[149] Vibhor Rastogi, Ashwin Machanavajjhala, Laukik Chitnis, and Anish Das Sarma. Finding connected components in map-reduce in logarithmic rounds. *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013.* Pp. 50–61.

[150] Eliot W. Robson, Jack Spalding-Jamieson, and **Da Wei Zheng**. Carving Polytopes with Saws in 3D. *Proceedings of the 36th Canadian Conference on Computational Geometry, CCCG 2024, Brock University, St. Catherines, Ontario, Canada, July 17 - July 19, 2024.* Pp. 145–151.

[151] Liam Roditty. A faster and simpler fully dynamic transitive closure. *ACM Trans. Algorithms.* 4 (2008). 6:1–6:16.

[152] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013.* Pp. 515–524.

[153] Liam Roditty and Virginia Vassilevska Williams. Minimum Weight Cycles and Triangles: Equivalences and Algorithms. *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011.* Pp. 180–189.

[154] Liam Roditty and Uri Zwick. A Fully Dynamic Reachability Algorithm for Directed Graphs with an Almost Linear Update Time. *SIAM J. Comput.* 45 (2016). Pp. 712–733.

[155] Liam Roditty and Uri Zwick. Improved Dynamic Reachability Algorithms for Directed Graphs. *SIAM J. Comput.* 37 (2008). Pp. 1455–1471.

[156] Tim Roughgarden, Sergei Vassilvitskii, and Joshua R. Wang. Shuffles and Circuits (On Lower Bounds for Modern Parallel Computation). *J. ACM.* 65 (2018). 41:1–41:24.

[157] Piotr Sankowski. Dynamic Transitive Closure via Dynamic Matrix Inverse (Extended Abstract). *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings.* Pp. 509–517.

[158] Walter Schnyder. Planar graphs and poset dimension. *Order.* 5 (1989). Pp. 323–343.

[159] Raimund Seidel. On the All-Pairs-Shortest-Path Problem in Unweighted Undirected Graphs. *J. Comput. Syst. Sci.* 51 (1995). Pp. 400–403.

[160] Jack Spalding-Jamieson, Brandon Zhang, and **Da Wei Zheng**. Conflict-Based Local Search for Minimum Partition into Plane Subgraphs. *38th International Symposium on Computational Geometry (SoCG 2022).* Vol. 224. 72:1–72:6.

[161] William L. Steiger and Ileana Streinu. A Pseudo-Algorithmic Separation of Lines from Pseudo-Lines. *Inf. Process. Lett.* 53 (1995). Pp. 295–299.

[162] Sairam Subramanian. A Fully Dynamic Data Structure for Reachability in Planar Digraphs. *Algorithms - ESA '93, First Annual European Symposium, Bad Honnef, Germany, September 30 - October 2, 1993, Proceedings.* Vol. 726. Pp. 372–383.

[163] Endre Szemerédi and William T. Trotter. Extremal problems in discrete geometry. *Comb.* 3 (1983). Pp. 381–392.

[164] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM.* 51 (2004). Pp. 993–1024.