Project Summary

We propose to revisit classic problems in computational geometry from the modern algorithmic perspective of exploiting the bounded precision of the input. In one dimension, this viewpoint has taken over as the standard model of computation, and has led to a powerful suite of techniques for both upper and lower bounds that constitute a mature field of research. But the same tools simply do not apply to two and higher dimensions, aside from orthogonal problems. For example, it is tempting to wonder whether the simple idea of radix sort can be extended to improve the running time of a simple two-dimensional problem like constructing a Voronoi diagram, but this problem has remained uncracked for many years despite extensive effort.

The investigators have jump-started this area in the past year with four papers [33, 18, 10, 6]. These results constitute the first compelling examples of how bounded precision can improve the running times of algorithms and data structures for fundamentally two-dimensional and nonorthogonal problems, such as planar point location, Voronoi diagrams, and dynamic convex hulls. More generally, this work initiates the development of a suite of techniques for exploiting bounded precision in geometric problems, laying the foundations for this new research direction.

Intellectual merit. The proposed research aims to fill the current gap between computational geometry, which generally assumes infinite precision inputs and computation, and most of the rest of theoretical computer science. This gap has become increasingly apparent as techniques for finite-precision one-dimensional problems have been improved and developed. For this reason, several researchers have exploited finite precision in orthogonal problems in higher dimensions, where essentially the same one-dimensional ideas apply, but have so far been unsuccessful in extending these results to nonorthogonal problems. This shortcoming for nonorthogonal problems was highlighted 15 years ago by Willard [45] and repeatedly since.

The fusion between computational geometry and fixed-precision computation has intellectual merit for both fields. In computational geometry, our goal is to elucidate the fundamental ways in which geometric information such as points and lines can be decomposed in algorithmically useful ways, enabling a deeper understanding of the relative difficulty of geometric problems. The results so far open a whole new playing field where most geometric problems do not yet have optimal solutions, and we propose to find such solutions. From the one-dimensional perspective, our goal is to replace the existing approaches for exploiting fixed precision with new techniques that generalize to multiple dimensions. By analogy with one dimension, it is reasonable to expect an equally rich theory for multidimensional problems.

Beyond this new area, lower bounds for geometric problems in fixed-precision models of computation should provide new insights into the original infinite-precision geometric problems. The absence of geometric lower bounds in fixed-precision models of computation opens a new research program for lower bounds in computational geometry. By renewing our understanding of classic geometric problems from the information-theoretic perspective, we aim ultimately to tackle geometric problems that are unsolved even in infinite-precision models and which seem far beyond the reach of current techniques. In the orthogonal context, work by the investigators has already illustrated this phenomenon with the partial-sums problem [35] and two-dimensional range counting [34], where major open problems in infinite-precision models could be solved only after the development of appropriate information-theoretic tools in the fixed-precision context.

Broader impact. There is a close relation between the proposed research and engineering practice. Geometric computer programs already exploit the fixed precision of the input, and the investigators' initial work in this area can be seen as a justification for some of these "heuristics." Our goal is to understand the extent to which fixed precision can be exploited and to develop new, more practical techniques for solving classic geometric problems.

The proposed research is a synergistic activity between traditional computational geometry and modern algorithms and data structures. This approach will lead to a more fluent exchange of ideas between the two fields. The investigators are also working to integrate the educational curricula of the two disciplines through this common ground.

Project Description

1 Introduction

The proposed project will revisit classic problems in computational geometry from the modern algorithmic perspective of bounded-precision input and computation. This research will fill the void of techniques for exploiting this bounded precision in nonorthogonal geometric algorithms and data structures. These problems have been explicitly studied for at least 15 years [45], but no progress has been made until the past year. This recent breakthrough by the investigators naturally opens the door for a new research program with tantalizing open questions that are the subject of this proposal.

Preliminary research. The investigators have jump-started this area with four main papers:

1. In FOCS 2006 [33], we develop the first sublogarithmic bound for planar point location queries. Specifically, our linear-space static data structure supports queries in $O(\min\{\lg n/\lg\lg n, \sqrt{w/\lg w}\})$ time, where w is the number of bits of precision (word size). This result also constitutes the first sublogarithmic bound for exact nearest neighbors in the plane. Similar results were independently discovered by Chan [9], which has led to subsequent collaboration.
2. In SoCG 2007 [18], we develop the first dynamic data structure for a nonorthogonal geometric problem with sublogarithmic query time. Specifically, we show how to maintain the convex hull of a set of n points in polylogarithmic update time and $O(\lg n/\lg\lg n)$ query time. We also prove a matching lower bound.
3. In [10], we develop an even faster algorithm for offline planar point location, running in time $n \cdot 2^{O(\sqrt{\lg\lg n})}$, which is $o(n \lg^{\varepsilon} n)$ for all $\varepsilon > 0$. This basic result enables many fundamental algorithmic problems to be solved in the same time bound: Voronoi diagram construction, Delaunay triangulation, 3D convex hull, line-segment intersection reporting, triangulating a polygon with holes, largest empty circle, Euclidean minimum spanning tree, etc.
4. In Algorithmica [6], we develop the first subquadratic algorithms for 3SUM, a problem which plays an important role in understanding geometric problems that seem to require superlinear polynomial time.

Indirectly, our work on the predecessor problem [37, 38], combined with a reduction of Chan [8], resolves the complexity of the approximate nearest neighbor problem in $O(1)$ dimensions.

Research team. Demaine is a leader in computational geometry and data structures. Pătraşcu is a leader in data structures with a speciality in lower bounds. Both investigators have a strong track record of solving long-standing open problems in these fields. For example, their joint work on logarithmic lower bounds [35] resolves the complexity of fundamental data structural problems in the powerful cell-probe model, solving multiple 15-year-old open problems. Their joint work [17] obtains the first $o(\lg n)$-competitive binary search tree, making the first major advance on the 20-year-old dynamic optimality conjecture.

The investigators share the information-theoretic perspective that is central to this proposal. This perspective has already led to the significant results in bounded-precision geometry described above. These results in particular settle 15-year-old open problems posed by Willard [45], but more excitingly, open the door for a whole new research program that is the topic of this proposal.

Roadmap. Section 2 describes the models of interest, why bounded precision is an important consideration, and the intellectual merit of geometric problems in this context. Section 3 describes

the key techniques that we have developed so far for solving these problems. Sections 4–6 describe the major open problems that remain in geometric algorithms, static data structures, and dynamic data structures, respectively. Section 7 highlights the broader impact of the proposed research. Section 8 summarizes results from prior NSF support.

2 Overview of Intellectual Merit

In this section we enumerate the several motivations for considering geometric problems in the bounded-precision context:

2.1. Real computers and therefore their inputs have bounded precision.
2.2. Bounded precision is already exploited in practice.
2.3. The fundamental problems under consideration have been studied and optimized extensively.
2.4. Exploiting bounded precision for these problems has been posed and attempted extensively.
2.5. New, faster algorithms may impact engineering practice.
2.6. The problems require the development of new, more powerful techniques of independent theoretical interest.
2.7. Progress in the bounded-precision context has a history of enabling solution to classic infinite-precision problems.

2.1 Theory for Reality

A central issue in computational geometry is the discrepancy between the idealized *geometric view* of limited objects with infinite precision, and the realistic *computational view* that everything is represented by (finitely many) bits.

The geometric view is inspired by Euclidean geometric constructions from circa 300 BC. In computational geometry, this view is modeled by the real RAM and related models considered for lower bounds, e.g., linear/algebraic decision/computation trees. These models postulate a memory of infinite-precision cells, holding real values, and specify the operations that can be performed on these values (typically, a subset of addition, multiplication, division, roots, and comparisons). Inputs and, depending on the problem, outputs consist of real values.

The computational view matches the reality of digital computers as we know them today and as set forth by Turing in 1936 [43]. This view assumes input is given with some finite precision, and that memory cells (words) have a precision comparable to the input. The preferred model for upper bounds is the word RAM, which allows operations commonly available in a programming language such as C, including bitwise operations. Lower bounds are usually shown in the cell-probe model, which allows *any* operation on a constant number of words. Thus, lower bounds have a deep information-theoretic meaning, and apply regardless of the exotic operations that might be implemented on some machine.

Traditionally, computational geometry has seen the negative side of the contrast between these two models. Algorithms are typically designed and analyzed in the real RAM, which makes the theoretical side easier. However, practitioners must eventually deal with the finite precision, making theoretical algorithms notoriously difficult to implement.

Given that algorithms must eventually be implemented in a bounded-precision model, it seems only natural not to tie our theoretical hands by using the real RAM. By recognizing that actual input data has bounded precision, and by designing algorithms for the word RAM, one could potentially obtain significantly better bounds. This ability has been demonstrated for some key problems by our preliminary work. In addition, this approach has the advantage that it does not

hide a large implementation cost by idealizing the model, and therefore has the potential to lead more directly to fast practical algorithms.

2.2 Theory for Practice

When scrutinizing a direction of theoretical research, it is important to understand not only whether it studies an aspect of reality, but also whether it studies an *interesting* aspect of reality. After all, the greatest payoff for a real application is often achieved by considering an appropriate abstraction of the object of study, not all the real but irrelevant details.

A common theoretical fallacy is that it is irrelevant to study algorithms in a bounded universe because "only comparison-based algorithms are ever implemented". However, this thesis has been attacked forcefully in one dimension; see, e.g., [25]. It is well known, for instance, that the fastest solutions to sorting are based on bounded precision (radix sort). Furthermore, when search speed matters, such as for forwarding packets in Internet routers, implementing search by comparisons is inconceivable [16].

In nonorthogonal geometry, the object of our study, there are at least two classes of examples showing the benefit of using bounded precision.

First, as discussed in a survey by Snoeyink [41], the most efficient and popular approaches for planar point location use pruning heuristics on the grid. These heuristics are similar in spirit to the algorithms we have developed for the problem [33], yielding the first sublogarithmic bounds. To some extent, our work justifies the advantage of these algorithms compared to the traditional approaches taking logarithmic time, which have been regarded as optimal.

Second, there is extensive study and implementation of the approximate nearest neighbor problem, even in two dimensions; see, e.g., [1, 8]. This is hard to understand when viewed through the real-RAM abstraction, because the exact nearest neighbor problem should be equally hard, both taking logarithmic time. However, the approximate version turns out to be equivalent to one-dimensional predecessor search. When dealing with bounded precision, this problem admits an exponentially better solution compared to exact search in two dimensions—$O(\lg w)$ versus $O(\sqrt{w/\lg w})$, where w is the number of bits of precision.

Thus, some of our work so far can already be seen as a theoretical justification for practically proven approaches. Though giving a theoretical understanding of practical algorithms is not our foremost goal, it is a normal and important outcome of theoretical research, in line with a significant body of modern work, such as justifying the simplex algorithm through smoothed analysis.

We should note, however, that not all aspects of engineering practice are yet understood. For example, practical implementations of grid-based search typically work with the original instance of planar point location. By contrast, our theoretical analysis assumes that the input is first reduced to the slab case, which incurs a constant but practically significant overhead.

2.3 The Problems in Historical Context

The problems that we have studied or propose to study (see Sections 4–6 on open problems below) are some of the most fundamental in computational geometry. Problems like planar point location or constructing Voronoi diagrams appear in virtually every textbook and are the topic of numerous surveys. Interest in these problems has not waned throughout the years, and new techniques have been developed continuously.

Taking planar point location as an example, we note that there are at least five fundamentally different ways to achieve $O(\lg n)$ query time with $O(n)$ space: planar separators [29], persistence [14, 39], triangulation refinement [28], separating chains plus fractional cascading [19], and randomized

incremental construction of a trapezoidal decomposition [31]. Taking this bound even further, Seidel and Adamy [40] obtain a running time of *exactly* $\log_2 n + 2\sqrt{\log_2 n} + O(\sqrt[4]{\log_2 n})$ point–line comparisons, with expected linear space.

In recent years, there has been a lot of interest in obtaining adaptive (but still comparison-based) bounds for point location, which can sometimes be sublogarithmic. The setup assumes queries are chosen from a biased distribution of entropy H, and one tries to relate the query time to H. Following some initial work on the subject, SODA 2001 saw no less than three results in this direction: Arya et al. [5] and Iacono [26] independently achieve expected $O(H)$ comparisons with $O(n)$ space, while Arya et al. [4] achieves $H + o(H)$ comparisons but with $O(n \lg^* n)$ space.

Historically, such directions have also been pursued intensively in one dimension (e.g., static and dynamic optimality). Some central problems in this field remain open. Despite this, almost two decades after Fredman and Willard's fusion trees [22], it appears that bounded precision safely occupies the center stage, and has grown into a much larger and more cohesive theory.

2.4 The Model in Historical Context

The idea of benefitting from bounded precision is certainly not new, even in computational geometry. As early as SODA 1992, less than two years after fusion trees were introduced, Willard [45] used fusion trees to improve bounds for certain geometric problems of an orthogonal nature, and asked about nonorthogonal problems, such as constructing Voronoi diagrams. Orthogonal problems, which naturally decompose into one-dimensional problems, have been studied with great success since then, with dozens of papers published. By contrast, his question about constructing Voronoi diagrams stood unanswered prior to our work in FOCS 2006 [33].

In isolated cases, nonorthogonal problems have been solved by reductions to one-dimensional problems. For example, convex hull reduces to sorting by the gift-wrapping algorithm. More recently, Chan [8] has shown that approximate nearest neighbor and a few other approximate distance problems reduce to one-dimensional searching or sorting. In these cases, understanding in the one-dimensional world immediately translates to the geometric world.

For most interesting problems, however, such reductions do not seem to exist. A popular alternative is to try to adapt some well-understood solution from integer search in one dimension, usually the structure of van Emde Boas [44]. However, this search strategy divides the problem in a way that is quite foreign to its inherent geometry. Thus, the space bounds become prohibitively high in the worst case, because this subdivision may break a segment into a lot of pieces. Illustrative examples of this philosophy are found, for example, in the works of Amir et al. [1] or Iacono and Langerman [27]. These papers achieve an $O(\lg \lg u)$ query time for point location. However, their space complexity is only bounded by measures such as the quad-tree complexity or the fatness, which could lead to prohibitive space bounds for difficult input instances.

Our proposed research aims to fill in this large void by developing the fundamentally new techniques that nonorthogonal problems need. We believe our recent work, addressing such diverse and fundamental problems as planar point location, Voronoi diagrams, and dynamic convex hull, give compelling examples of how bounded precision can lead to both important improvements, and fascinating theoretical questions. This should provide a sufficient impetus for accelerated developments in the field, hopefully catching up with the high degree of sophistication achieved in the one-dimensional case.

2.5 Practice for Theory

We firmly believe that the relation between theory and practice should be bidirectional, and promising theoretical ideas should find their way into practical systems. We believe our work can have a

practical impact for two reasons.

First, as noted above, algorithms of the same flavor (e.g., gridding heuristics) are already used successfully in practice. Mathematically founded ideas for grid search could lead to better practical algorithms.

Second, for the algorithmic problems we have considered (e.g., constructing Voronoi diagrams), the theoretical improvement over classic solutions is rather significant—$o(n \lg^{\varepsilon} n)$ versus $O(n \lg n)$. In the similar case of sorting, it is widely appreciated that algorithms based on bounded precision (radix sort) outperform $O(n \lg n)$ sorting algorithms.

Unfortunately, we have not yet been able to test the practicality of the new ideas from our work. NSF support would allow us to better disseminate these ideas to experimental circles, as well as fund an undergraduate researcher who would implement and test these algorithms.

2.6 Theory for Theory

All things considered, it is perhaps the theoretical view of this emerging field that we find most compelling. It is a general phenomenon that tools developed in one dimension for exploiting bounded precision seem foreign to two or more dimensions, and simply do not help. As such, our results so far have had to develop interesting new techniques, rebuilding our understanding of bounded precision from scratch. By analogy with the one-dimensional case, a mature research field with many dozens of publications, we can hope that in the multidimensional case lurks a similarly rich theory.

A similar account can be made from the geometric side. Our work so far has forced us to re-analyze many "standard" ideas and techniques in geometry through a new, information-theoretic lens. This view of geometry seems interesting in its own right, and the algorithmic puzzles it raises are very appealing.

Below, we dedicate the entire Section 3 to highlighting some of the key techniques known to date, contrasting them to one-dimensional techniques.

2.7 Merit beyond Bounded Precision

We have argued above that geometry with bounded precision is both a relevant and interesting topic of research. Stepping back for a moment, we argue that there are compelling reasons for the proposed study far outside the realm of bounded precision.

An overwhelming number of fundamental problems in computational geometry are far from understood even in classic models of computation. For example, in the real RAM, or even in the simpler group model, we do not know how to prove *any* lower bound exceeding $\Omega(\lg n)$ for a static data structure (and often not even that).

We believe progress even in these models requires a firm information-theoretic understanding of the problems. A good strategy seems to start developing such lower bound tools from the ground up, starting with smaller lower bound for the problems where bounded precision helps. Though these lower bounds are asymptotically small, they are forced to be in the right spirit of understanding geometry through information. Starting with these computationally easier problems provides a natural program for gradual development of increasing lower bounds.

A compelling example of this philosophy is our recent breakthrough in group-model lower bounds for orthogonal range queries [34]. This work addresses important questions in classic computational geometry, posed, for example, by Fredman in JACM 1982 [21] and Chazelle in FOCS 1986 [11]. However, this result came as a culmination of our work in the cell-probe model [37, 36, 38]. This

line of work had to develop key information-theoretic tools in trying to understand the predecessor problem, the totem of one-dimensional search with bounded precision.

3 Key Techniques

In this section, we discuss the main techniques developed by our initial investigations in the area of finite-precision geometry, contrasting to traditional one-dimensional techniques. We keep the discussion at a general level, focusing on an information-theoretic view of the algorithmic techniques. Details about applications to specific problems are delayed to Sections 4–6 which emphasize open problems.

3.1 On Sketching

One of the most influential ideas in one-dimensional search is given by fusion trees [22]. This structure shows how to sketch $B = w^{\varepsilon}$ words, each w bits long, into a single word of w bits such that a predecessor query among the B numbers can be answered in constant time after seeing the sketch. Thus, to find the predecessor among n numbers, one simply builds a balanced B-tree with such a sketch at every node. A query walks down a root-to-leaf path in constant time per node.

For planar point location, we suspect that this type of sketching is impossible for any superconstant B. Thus, our sublogarithmic query algorithm [33] had to be based on a novel approach. To best understand this idea, we define an information theoretic measure for a subregion in a vertical slab. Refer to Figure 1. The key observation is that, while we may not be able to efficiently locate the query point among *any* B segments, we can do so as long as the region between segments does not have too small entropy.

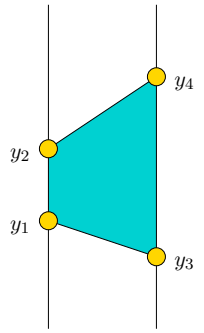

Figure 1: $H = \lg(y_2 - y_1) + \lg(y_4 - y_3)$

More specifically, the search starts by assuming that the query point can lie anywhere in a slab, a region with high entropy. By appropriate sketching or tabulation techniques, each step of the algorithm reduces the entropy of the region where the query could lie, by some fixed increment ΔH. The answer is certainly found when the entropy of the region becomes small enough, so we can bound the query time in terms of the entropy progress in each step. We note that information-progress arguments of this flavor are rather common in lower bounds, but generally their use in designing algorithms has not been widely appreciated.

We observe that this search strategy behaves quite differently from fusion trees. For instance, the $O(\log_w n)$ bound of fusion trees means the problem actually gets *easier* for very high precision. For point location, on the other hand, it is not known how to improve the $O(\lg n / \lg\lg n)$ bound for any high precision.

3.2 On Bucketing

Bucketing is another fundamental technique in one dimension, used pervasively for reducing space bounds and update times of dynamic data structures. The idea is to bucket, say, $\Theta(t)$ consecutive elements, building a bottom structure inside each bucket, and a top structure for elements separating the $\Theta(n/t)$ buckets.

A remarkable application of this idea are the exponential trees of Andersson and Thorup [3]. Starting with a static data structure using space, say, $O(n^2)$, they give a linear-space data structure, which is even dynamic! To see this, imagine choosing $t = n^{2/3}$. Then, the top structure uses space $O((n/t)^2) = O(n^{2/3})$, while the bottom structures use space $\frac{n}{t} \cdot O(t^2) = O(n^{4/3})$. This is already an

improvement over the original quadratic space, and recursing on the bottom structures takes the space down to linear. Note that this structure will even be dynamic, because each top structure needs to be rebuilt very infrequently compared to its size: for the first level, once in $\Theta(\frac{n}{t}) = \Theta(n^{2/3})$ operations.

In two dimensions, the equivalents of bucketing and exponential trees seem very difficult to achieve. Indeed, this would have important applications even for unbounded precision: it would imply logarithmic upper bounds for *dynamic* point location, a famous open problem. The difficulty in bucketing comes from the apparent impossibility to choose good separators. Say, for instance, that some segment s_1 is chosen to separate two buckets. In the very next step, s_1 could be deleted, and another s_2 which intersects s_1 can be inserted. Now, unless we rebuild the top structure to replace s_1 (a very expensive operation), we do not know in which bucket to place s_2.

A problem faced by our static point location structure is that the direct implementation has superlinear space. In the one-dimensional case, this problem would be easily fixed by exponential trees. In two dimensions, we were still able to achieve linear space by observing that our application only needed to construct exponential trees in the *semionline* case (for every element, we are told the time in the future when it will be deleted). Motivated by this property, we constructed semionline geometric exponential trees by a careful interplay of geometric separation arguments and standard amortization arguments.

3.3 The Dynamic Challenge

Dynamic data structures present some of the toughest challenges in our study. As opposed to static data structures, where we are free to organize a hierarchical structure based on our information-progress argument, dynamic data structures already need a different hierarchical structure for the sake of dynamic maintenance. Quite expectedly, the requirements of these two structures can clash in catastrophic ways.

This phenomenon is immediately apparent for dynamic convex hull, considered in some of our preliminary work [18]. When starting with a classic maintenance strategy of Overmars and van Leeuwen [32], which supports updates in $O(\lg^2 n)$ time, we were able (with careful modifications) to preserve the information progress along a search path in this structure. This led to an $O(\frac{\lg n}{\lg \lg n})$ query time. For another dynamic maintenance strategy of Chan [7], which has the advantage of near-logarithmic update times, we were generally unable to achieve a similar result, with the exception of some restricted queries.

Preserving the information progress in the structure of Overmars and van Leeuwen involves challenging geometric arguments that led to an understanding of this structure from entirely new and unexpected points of view. It is a fascinating open question whether one can achieve the needed information-preserving geometry in other structures that surpass the $O(\lg^2 n)$ update time.

3.4 Sorting in Two Dimensions

Most of our algorithmic results are derived from an algorithm for the *offline* point location problem, which is a natural extension of sorting to two dimensions (sorting points with respect to lines).

Although we are dealing with a generalization of one-dimensional sorting, existing techniques from integer sorting do not seem to help. Consider, for example, two standard tools in sorting. One is radix sort, which gives linear-time sorting in polynomial universes, i.e., when $w = O(\lg n)$. Another is hashing, which is used in virtually all advanced RAM sorting algorithms. Hashing, in the form of signature sort [2], gives linear-time sorting when $w \geq \lg^{2+\varepsilon} n$. Thus, sorting takes linear time both for appropriately small and large precision.

In two dimensions, neither of these two tools seem to apply. The existence of two-dimensional radix sort or hashing is a tantalizing question that we have not been able to resolve. It is interesting that, even without such basic tools, we [10] were still able to obtain a rather efficient, deterministic algorithm, even outperforming some of the older RAM sorting results. Our algorithm achieves a running time of $n \cdot 2^{O(\sqrt{\lg \lg n})}$ regardless of the precision, and it is not known how to improve this for very small or for very large precision.

As before, our sorting strategy came from consideration of how information can be manipulated algorithmically. Say we have some points and segments in a region of entropy H. By an appropriate geometric argument, we pick some subset of the segments, and rescale to a coarser grid in which the region has entropy H'. Then, based on a solution to this problem, we are able to break the original problem into subproblems of entropy $H - H'$. This geometric technique leads to interesting sorting strategies in which, at various moments in time, points are repacked more tightly in a word, using the reduced entropy, in order to speed up a scan of the points. This information-manipulation problem poses rather unusual challenges that do not appear to be shared with any one-dimensional problem.

4 Open Problems in Algorithms

We now turn to a collection of fundamental problems in computational geometry that we propose to examine in the information view because we believe it will reveal additional structure on the problem in particular and geometric information in general. We divide the open problems into three major categories: *algorithms* (compute an entire structure such as the Voronoi diagram, or bulk offline queries), *static data structures* (preprocess the geometry to support fast queries), and *dynamic data structures* (maintain geometry subject to fast updates and queries). Computational geometry is ripe with relations between such categories of problems: many data structures have been invented solely because they resulted in an optimal geometric algorithm. But under this new vantage point, we need to reconsider whether such reductions still produce optimal solutions, or whether additional structure and separations result.

We begin in this section with algorithmic open problems. Equivalently, we can view such problems as data structural, but all of the data and the queries are given upfront (offline), and the goal is to answer all queries in bulk.

4.1 Offline Planar Point Location

One of the most basic algorithmic problems in computational geometry is *offline (bulk) planar point location*: given a planar map and a set of points, compute which face contains each point. This problem arises directly in applications such as finite-element simulations with Euler-Lagrange coupling [12],[1] and is an essential piece of overlay problems in GIS and CAD systems [15]. Offline point location is also the typical second half of the offline nearest-neighbor problem: given a set of sites and a set of query points, find the nearest site to each query point. (Later we will consider the first half: computing the Voronoi diagram.)

Our current best solution to this problem, just recently developed with Chan [10], runs in $n \cdot 2^{O(\sqrt{\lg \lg n})}$ time, where n is the number of points plus the combinatorial complexity of the planar map. This time bound is so far the fastest (per point) obtained for any problem in this context, being $n \cdot o(\lg^{\varepsilon} n)$ for all $\varepsilon > 0$, but it seems unlikely that the bound is optimal.

[1] Roughly speaking, the Eulerian mesh is a planar map (representing points in space), while a Lagrangian mesh is often a cloud of point particles (representing the fluid's point of view). At every simulation time step, particles must be localized within its Euler element.

Research Challenge 1: Surpassing $2^{\sqrt{\lg \lg n}}$. Is there an $n(\lg \lg n)^{O(1)}$-time algorithm for offline planar point location? Does randomization help?

One approach to this problem is to consider randomized algorithms. A faster randomized algorithm would be particularly interesting as it would be essentially the first improvement through randomization of a geometric problem in the information view. There is ample precedent for randomization offering simpler optimal solutions to geometric problems, such as randomized incremental construction, but randomization also plays an important role in the current state-of-the-art in integer sorting [25]. Therefore we are hopeful that randomization can further improve this relatively new algorithm.

On the other hand, it seems unlikely that planar point location can be solved in linear time. In particular, it can be shown that it is at least as hard as integer sorting, for which the best algorithm runs in $O(n\sqrt{\lg \lg n})$ time [25] but no superlinear lower bound is known. Thus, likely the only hope for lower bounds in this context is to find a natural restriction on the model, perhaps obtained by examining the common features between sorting and planar point location, and proving a lower bound under that assumption. Such an approach may shed new light on the classic integer sorting problem.

4.2 Voronoi Diagrams

We obtain the same time bound, $n \cdot 2^{O(\sqrt{\lg \lg n})}$, but in expectation, for computing the Voronoi diagram on n points in the plane, or more generally for computing the convex hull of n points in 3D [10]. The idea is to use the random-sampling approach of Clarkson and Shor [13], refined in this context by Chan [9], to reduce a Voronoi diagram or more generally a 3D convex hull to the offline planar point location problem. Roughly, we take a random sample of $n/\lg n$ points, compute their hull, use offline point location to find the hull facet containing each of the n points in projection, compute a local hull of points in the same facet, and then patch together these local hulls.

Although frequently used in tandem with point location, Voronoi diagrams and 3D convex hulls are important structures in their own right, and it is natural to wonder whether they are in fact easier problems than offline planar point location. The additional structure offered by Voronoi diagrams may make the problem easier to solve. Alternatively, and perhaps more likely, we may be able to find a reverse reduction, converting any offline planar point location problem into an equivalent 2D Voronoi or 3D convex hull problem. Intuitively, the Voronoi diagram of the planar map vertices and the query points contains enough information to solve offline planar point location, but the details remain to be worked out. (If we could include the planar map's edges in the Voronoi diagram, the result would be clear, but a Voronoi diagram of points would be much more interesting.)

Research Challenge 2: Voronoi vs. Point Location. Is computing a Voronoi diagram as hard as offline planar point location?

Another direction to pursue is whether the random-sampling reduction from 2D Voronoi and 3D convex hull to offline planar point location can be made deterministic via derandomization. Unless we find a faster solution to offline planar point location that exploits randomization, it is unsatisfying to require randomization just for this reduction. Instead of derandomizing the reduction directly, an alternative is to attempt to apply our point-location techniques directly to constructing Voronoi diagrams, and therefore obtain a deterministic time bound. Although certainly difficult, we believe that this approach should be surmountable.

Research Challenge 3: Deterministic Voronoi. Can Voronoi diagrams be computed in $n \cdot 2^{O(\sqrt{\lg \lg n})}$ time without randomization?

4.3 Line-Segment Intersection

A more challenging algorithmic problem is *red-blue intersection counting*: given n line segments, each marked red or blue, count the number of intersections between red and blue. It is guaranteed that red, respectively blue, segments do not intersect among themselves. This problem can be solved in $O(n \lg n)$ time, and it is the only problem of this complexity, which is typically considered, and for which our techniques do not offer any improvement.

Note, however, that in this case we cannot hope to achieve bounds similar to constructing Voronoi diagrams. This problem is at least as hard as counting inversions in a sequence of integers, for which it is a long-standing open question to obtain a running time better than $O(n \lg n / \lg \lg n)$. Unfortunately, we do not know how to mimic this running time for the geometric problem.

Research Challenge 4: Counting Faster. Is there a $o(n \lg n)$-time algorithm for red-blue line-segment intersection counting?

Note that the same problem can be easily solved in the case when each line segment is either horizontal or vertical, by using an optimal partial-sums data structure [35].

4.4 Degeneracy Detection

A problem seemingly requiring larger polynomial time bounds is *degeneracy detection*: given n points in the plane, are any three of them collinear? This problem was the motivation for Gajentaan and Overmars [23] to introduce the 3SUM problem: given n integers, do any three of them sum to zero? Degeneracy detection is 3SUM-hard, meaning that there is a subquadratic reduction from 3SUM to degeneracy detection. The best known algorithm for either problem is $\Theta(n^2)$ in any model of computation with bounded branching, and indeed Erickson [20] proves quadratic lower bounds for restricted linear decision trees.

Exploiting that the inputs are integers, however, we have shown that 3SUM can be solved faster [6], in $O\left(n^2/\frac{w^2}{\lg^2 w}\right)$ time. This time bound represents a roughly quadratic speedup in the parallelism afforded by the model, which matches the seeming quadratic behavior of the problem. However, this result leaves open the original motivating problem of degeneracy detection. In ongoing work with Chan, we are pursuing a fusion between the 3SUM speedups with the various geometric tools to obtain analogous results for degeneracy detection. We expect that this fusion will be successful within the next few months.

Research Challenge 5: Subquadratic Degeneracy Detection. Is there a $o(n^2)$-time algorithm for degeneracy detection?

5 Open Problems in Static Data Structures

Next we turn to data structural open problems where the data is static but queries come online. Because we need an instant answer for each query, we cannot expect the same kind of speedup as for bulk queries. Indeed, here there are many more interesting questions about lower bounds, building on the field of (one-dimensional) data structural lower bounds.

5.1 Planar Point Location

The data structural form of planar point location is even more common than bulk queries. For example, in graphical user interfaces, we must determine the region in which the user clicked. Combining results of ours [33] and of Chan [9], we obtain a linear-space static data structure supporting point-location queries in $O(\min\{\lg n/\lg\lg n, \sqrt{w/\lg w}\})$ time. The important question here is whether these bounds can be improved:

> **Research Challenge 6: Static Planar Point Location.** How fast can point-location queries be supported by a linear-space data structure on n points?

Improving the present results would require the development of new techniques in finite-precision geometric algorithms. On the other hand, the present bounds are fairly natural, and may well turn out to be the correct answer. In this case, we would aim to find a matching lower bound. Such a lower bound would be particularly interesting because it would be fundamentally geometric, being the first lower bound for static data structures that is not an immediate consequence of known one-dimensional lower bounds. Specifically, the point of comparison here is the predecessor problem (the one-dimensional analog of point location), where we recently established the optimal query bound for a linear-space data structure to be $\Theta(\min\{\lg n/\lg w, \lg w\})$ [37]. This bound differs substantially from our point-location bound in the second term, roughly a square root versus a logarithm.

A somewhat easier question in this context is whether we can prove that standard techniques for the predecessor problem, such as the sketching approach of fusion trees, cannot generalize to the multidimensional setting (in some model). For example, it seems that there is no data structure supporting planar point location queries on a planar map of combinatorial complexity $\lg^{\varepsilon} n$ in constant time, thus making impossible the black box of a fusion-tree node.

5.2 Nearest Neighbor

The data structural analog of Voronoi diagrams is the *nearest-neighbor problem*: preprocess n point sites subject to queries for the site nearest a given point. In traditional computational geometry and the algebraic computation tree model, this problem requires $\Omega(\lg n)$ time in the worst case, so it is optimal to reduce this problem to planar point location. But from the finite-precision perspective, we may be able to circumvent any lower bounds for planar point location in the special case of finding nearest neighbors. Of course, the lower bounds of the predecessor problem still apply here, but it is possible that the difficulty of nearest neighbor falls in between predecessor and planar point location.

> **Research Challenge 7: Static Nearest Neighbor.** How fast can nearest-neighbor queries be supported by a linear-space data structure on n points?

Alternatively, we may establish lower bounds for nearest neighbor stronger than predecessor. For example, if we can establish such strong lower bounds for planar point location, the argument may be transferable to the special case of nearest neighbors. If we obtain such lower bounds, we will obtain the first formal sense in which *approximate* nearest neighbors, which reduces to a one-dimensional predecessor problem [8], is asymptotically easier to solve than exact nearest neighbors. This theoretical result would justify the observation in practice that approximate nearest neighbor is substantially easier than exact nearest neighbor, even in the plane.

5.3 Connectivity

A subtle variation on planar point location is *planar point connectivity*: preprocess a planar map subject to queries for whether two given points are in the same face of the map. Obviously, this problem reduces to planar point location: determine the face containing the two points and test for equality. But it is possible that planar point connectivity can be solved faster than not only planar point location, but possibly even the predecessor problem. The analogy here is to one-dimensional range reporting (is there a point in a given interval?), which we recently showed to be solvable exponentially faster than the predecessor problem [30].

> **Research Challenge 8: Planar Point Connectivity.** Can static data structures for planar point connectivity outperform those for planar point location?

This problem arises in a practical setting introduced by Thorup [42]. Specifically, an efficient data structure for connectivity is essentially what is needed to obtain efficient approximate distance oracles. Here we would like to know the approximate number of faces (dual vertices) along the best path between two query points in a static planar map.

5.4 Polygons

On the simpler side, we can imagine queries about just a single convex polygon in the plane. These queries naturally arise as static analogs to queries studied previously in the dynamic convex hull problem. A simple example can be phrased in a linear-programming context: preprocess a two-dimensional convex polytope to support queries for the vertex optimizing a given objective function. This problem is actually equivalent to predecessor, so it is not interesting theoretically. Another easy query is testing whether a given point is inside the polygon, which can be solved with a predecessor query in a persistent structure.

Other queries are less trivial, however. A *tangent query* asks for a tangent to the convex polygon through a given point. A *line-stabbing query* asks for the two edges of the polygon struck by a given line. We suspect that neither of these problems can be solved as quickly as the predecessor problem.

> **Research Challenge 9: Static Polygons.** How fast can tangent queries or line-stabbing queries be supported by a static data structure on a convex n-gon?

6 Open Problems in Dynamic Data Structures

Our final collection of open problems is also the most challenging: maintain geometric data subject to both queries and updates. This direction of research is one we have only just begun to explore [18], and it seems the richest. The techniques for static data structures do not generally apply, requiring the development of new techniques.

6.1 Convex Hull

Our results for dynamic convex hull [18] suggest a new discrepancy between different types of queries. Specifically, we show that hull membership and linear-programming queries can be supported in $O(\lg n/\lg\lg n)$ time while point insertion and deletion take only $O(\lg n \lg\lg n)$ time. On the other hand, we show how to support all previously considered queries (including, e.g., tangent queries and line-stabbing queries) in $O(\lg n/\lg\lg n)$ time while point insertion and deletion take $O(\lg^2 n)$

time. In contrast, Brodal and Jacob's data structure with $O(\lg n)$ query and update time can solve tangent queries but not line-stabbing queries. Are these queries really different from each other?

> **Research Challenge 10: Dynamic Convex Hull Queries.** Is the optimal update time for an $O(\lg n/\lg\lg n)$ linear-programming query faster than for an $O(\lg n/\lg\lg n)$ tangent query? Is the latter faster than for an $O(\lg n/\lg\lg n)$ line-stabbing query?

Of course, another natural question is whether the updates can be improved to logarithmic or even sublogarithmic while maintaining fast updates. We do not know of any lower bounds that prevent, say, an $O(\lg\lg n)$ update time, although this seems unlikely. Resolving this issue will likely require the development of new geometric lower bound techniques.

> **Research Challenge 11: Dynamic Convex Hull Updates.** What is the optimal update time for dynamic convex hull while supporting $O(\lg n/\lg\lg n)$ linear-programming queries?

We also show in [18] an $\Omega(\lg n/\lg w)$ lower bound for all previously considered dynamic convex hull queries (except gift wrapping which requires only constant time). However, we have not been able to achieve that query bound. We believe that it may be possible, but a stronger lower bound would be even more interesting.

> **Research Challenge 12: Dynamic Convex Hull Updates.** Is the optimal query time for dynamic convex hull with polylogarithmic updates $O(\lg n/\lg\lg n)$ or $O(\lg n/\lg w)$?

6.2 Point Location

Our new bounded-precision perspective may shed new light on classic unsolved problems such as dynamic planar point location. The best classic data structure for this problem [24] supports queries in $O(\lg n\lg\lg n)$ amortized time and updates in $O(1)$ amortized time. A long-standing open question is whether these bounds can be improved to $O(\lg n)$ amortized time for all operations. In the finite-precision model, it seems plausible that we could improve by a $\lg\lg n$ factor. But more interestingly, we conjecture that we can improve updates and/or queries to be sublogarithmic. Such a result in the finite-precision model may lead in turn to better results in the infinite-precision model, which would be particularly exciting.

> **Research Challenge 13: Dynamic Planar Point Location.** Can dynamic planar point location be solved in logarithmic, or even sublogarithmic, time per operation?

6.3 Plane Graphs

Part of dynamic planar point location is the maintenance of a planar map subject to insertion and deletion of points and edges. Two other natural queries on such dynamic planar maps are (a) are two given edges on the same face? and (b) are two vertices in the same connected component of the graph? The latter query is a natural analog of dynamic connectivity in graphs. We have shown that

dynamic connectivity requires $\Omega(\lg n)$ time per operation, even when the graph is planar. However, our construction does not preserve the planar embedding of the graph; it can change drastically in each step. When the user must explicitly maintain the embedding by modifying the points that define vertices, the problem may be substantially easier.

Research Challenge 14: Dynamic Connectivity in Plane Graphs. Can dynamic connectivity in plane graphs be solved in sublogarithmic time per operation?

If the answer to this question is positive, we would have a nice contrast where the two-dimensional problem is actually easier than the motivating data structural problem: geometry helps. Even if dynamic connectivity is not possible in sublogarithmic time, the simper query of testing whether two edges bound the same face may be possible, cutting a finer line through the problem space.

7 Broader Impact

The broader impact of the proposed research is on two fronts: practice and education.

On the practical side, we have described throughout the proposal several heuristically observed phenomena that can now be justified by theoretical results in the finite-precision model. Some of these justifications already follow from our preliminary research, and many more would follow from solutions to open problems proposed here. Given these intrinsic connections to practice, we are hopeful that some of the techniques we develop will provide useful speedups in geometric computer programs. We will hire an undergraduate researcher to aid the exploration of the practicality of each technique we develop.

On the educational side, creating a bridge between the fields of computational geometry and finite-precision algorithms and data structures has several positive consequences. We anticipate that this bridge will create a common ground in which researchers from both sides can participate, fostering the exchange of ideas. We have already given several invited lectures about this new perspective, and have agreed to give a guest lecture in the MIT computational geometry class about this topic. We will also incorporate material into the MIT advanced data structures class which we codeveloped. In this way, we hope to educate a new generation of computer scientists that appreciate the interplay between geometry and finite precision and can help us build a better understanding of it.

8 Results from Prior NSF Support

We focus here on PI Demaine's primary NSF support, from CAREER grant CCF-0347776 on "Fundamental Research in Geometric Folding." This project has opened many new collaborations, including a deployable structures research team with Zhong You and John Ochsendorf; a protein folding team with Stefan Langerman, Joseph O'Rourke, and biology researchers at Gembloux; a folding simulation team with undergraduate Jenna Fizel; and an origami design team with Robert Lang. We also developed a new MIT class on geometric folding algorithms, taught in Fall 2004 and to be taught again in Fall 2007, which led to significant new research by both undergraduate and graduate students. For outreach, we have given lectures about geometric folding the Women's Technology Program (MIT, June 2006 and June 2005), at the Museum of Science (Boston, May 2005), and at the Annual Meeting of the American Association for Advancement of Science (AAAS, Washington, DC, February 2005). We have also given plenary lectures about geometric folding at the 4th International Meeting of Origami Science, Math, and Education (Pasadena, California,

September 2006), the AMS-IMS-SIAM Joint Summer Research Conference on Discrete and Computational Geometry—Twenty Years Later (Snowbird, Utah, June 2006), the Université Catholique de Louvain (Louvain-la-Neuve, Belgium, April 2006), the Joint Mathematics Meetings of the American Mathematical Society and Mathematical Association of America (Atlanta, Georgia, January 2005), the 15th Annual International Symposium on Algorithms and Computation (Hong Kong, China, December 2004), and the Computer Science Invitational Lecture Series, University of Waterloo, (Waterloo, Canada, October 2004). I also give guest lectures on geometric folding in two MIT classes, on introduction to computer science and on computational geometry.

Together with Joseph O'Rourke, we have completed our book *Geometric Folding Algorithms: Linkages, Origami, and Polyhedra*. The manuscript is now being typeset by Cambridge University Press, and will appear in May 2007. The 600-page manuscript represents the culmination of a significant effort in bringing together the topics of this research into one cohesive framework. In addition, the following related papers have been published during the span of this project.

1. "Polygons Flip Finitely: Flaws and a Fix" (with Blaise Gassend, Joseph O'Rourke, and Godfried T. Toussaint), in *Proc. 18th Canadian Conference on Computational Geometry*, 109–112, 2006.
2. "Locked and Unlocked Chains of Planar Shapes" (with Robert Connelly, Martin L. Demaine, Sándor Fekete, Stefan Langerman, Joseph S. B. Mitchell, Ares Ribó, and Günter Rote), in *Proc. 22nd Annual ACM Symposium on Computational Geometry*, 61–70, 2006.
3. "Refolding Planar Polygons" (with Hayley N. Iben and James F. O'Brien), in *Proc. 22nd Annual ACM Symposium on Computational Geometry*, 71–79, 2006.
4. "The Helium Stockpile: A Collaboration in Mathematical Folding Sculpture" (with Martin L. Demaine and A. Laurie Palmer), *Leonardo* 39(3):233–235, 2006.
5. "Puzzles, Art, and Magic with Algorithms" (with Martin L. Demaine), *Theory of Computing Systems*, volume 39, number 3, pages 473–481, June 2006. Special issue of selected papers from FUN 2004.
6. "Geometric Restrictions on Producible Polygonal Protein Chains" (with Stefan Langerman and Joseph O'Rourke), *Algorithmica* 44(2):167–181, 2006. Special issue of selected papers from ISAAC 2003.
7. "A Survey of Folding and Unfolding in Computational Geometry" (with Joseph O'Rourke), in *Combinatorial and Computational Geometry*, J. E. Goodman, J. Pach, and E. Welzl, eds., Mathematical Sciences Research Institute Publications 52, 167–211, 2005, chapter Cambridge University Press.
8. "Kinematics and Dynamics of Nanostructured Origami" (with Paul Stellman, Will Arora, Satoshi Takahashi, and George Barbastathis), in *Proc. ASME International Mechanical Engineering Congress and Exposition*, 541–548, 2005.
9. "Grid Vertex-Unfolding Orthostacks" (with John Iacono and Stefan Langerman), *International Journal of Computational Geometry and Applications*, to appear.
10. "Unfolding Polyhedral Bands" (with Greg Aloupis, Stefan Langerman, Pat Morin, Joseph O'Rourke, Ileana Streinu, and Godfried Toussaint), *Computational Geometry: Theory and Applications*, to appear.
11. "Hinged Dissection of Polyominoes and Polyforms" (with Martin L. Demaine, David Eppstein, Greg N. Frederickson, and Erich Friedman), *Computational Geometry: Theory and Applications* 31(3):237–262, June 2005. Special issue of selected papers from CCCG'99.
12. "When Can You Fold a Map?" (with Esther M. Arkin, Michael A. Bender, Martin L. Demaine, Joseph S. B. Mitchell, Saurabh Sethia, and Steven S. Skiena), *Computational Geometry: Theory and Applications* 29(1):23–46, 2004.
13. "Fold-and-Cut Magic" (with Martin L. Demaine), in *A Tribute to a Mathemagician*, 2004, A K Peters.
14. "Geometry and Topology of Polygonal Linkages" (with Robert Connelly), in *CRC Handbook of Discrete and Computational Geometry*, Second Edition, 197–218, 2004, chapter 9.
15. "Hinged Dissection of Polypolyhedra" (with Martin L. Demaine, Jeffrey F. Lindy, and Diane L. Souvaine), in *Proc. 9th Workshop on Algorithms and Data Structures*, 205–217, 2005.
16. "Folding Paper Shopping Bags" (with Devin J. Balkcom and Martin L. Demaine), in *Proc. 14th Annual Fall Workshop on Computational Geometry*, 14–15, 2004.

References Cited

[1] A. Amir, A. Efrat, P. Indyk, and H. Samet. Efficient regular data structures and algorithms for dilation, location, and proximity problems. *Algorithmica*, 30(2):164–187, 2001. See also FOCS'99.

[2] A. Andersson, T. Hagerup, S. Nilsson, and R. Raman. Sorting in linear time? In *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing*, pages 427–436, Las Vegas, Nevada, May–June 1995.

[3] A. Andersson and M. Thorup. Tight(er) worst-case bounds on dynamic searching and priority queues. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 335–342, Portland, Oregon, May 2000.

[4] S. Arya, T. Malamatos, and D. M. Mount. Entropy-preserving cuttings and space-efficient planar point location. In *Proc. 20th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 256–261, 2001.

[5] S. Arya, T. Malamatos, and D. M. Mount. A simple entropy-based algorithm for planar point location. In *Proc. 20th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 262–268, 2001.

[6] I. Baran, E. D. Demaine, and M. Pătraşcu. Subquadratic algorithms for 3SUM. *Algorithmica*. To appear.

[7] T. M. Chan. Dynamic planar convex hull operations in near-logarithmic amortized time. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 92–99, 1999.

[8] T. M. Chan. Closest-point problems simplified on the ram. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 472–473, San Francisco, California, 2002.

[9] T. M. Chan. Point location in $o(\log n)$ time, Voronoi diagrams in $o(n \log n)$ time, and other transdichotomous results in computational geometry. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 333–342, Berkeley, California, October 2006.

[10] T. M. Chan and M. Pătraşcu. Voronoi diagrams in $n \cdot 2^{O(\sqrt{\lg \lg n})}$ time. Submitted to STOC, 2007.

[11] B. Chazelle. Lower bounds for orthogonal range searching II. The arithmetic model. *Journal of the ACM*, 37(3):439–463, 1990. See also FOCS'86.

[12] T. J. Chung. Coupled eulerian-lagrangian methods. In *Computational Fluid Dynamics*, section 16.4.2, pages 525–528. Cambridge University Press, 2002.

[13] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.

[14] R. Cole. Searching and storing similar lists. *Journal of Algorithms*, 7(2):202–220, 1986.

[15] M. de Berg, H. Haverkort, S. Thite, and L. Toma. I/O-efficient map overlay and point location in low-density subdivisions. In *Abstracts from the 23rd European Workshop on Computational Geometry*, Graz, Austria, March 2007. To appear. `http://www.win.tue.nl/~sthite/pubs/overlay-eurocg07.pdf`.

[16] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink. Small forwarding tables for fast routing lookups. In *Proc. ACM SIGCOMM*, pages 3–14, 1997.

[17] E. D. Demaine, D. Harmon, J. Iacono, and M. Pătraşcu. Dynamic optimality—almost. *SIAM Journal on Computing*. To appear.

[18] E. D. Demaine and M. Pătraşcu. Tight bounds for dynamic convex hull queries (again). In *Proceedings of the 23rd Annual ACM Symposium on Computational Geometry*, Gyeongju, South Korea, June 2007. To appear.

[19] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.

[20] J. Erickson. Lower bounds for linear satisfiability problems. *Chicago Journal of Theoretical Computer Science*, 1999(8), 1999.

[21] M. L. Fredman. The complexity of maintaining an array and computing its partial sums. *Journal of the ACM*, 29(1):250–260, 1982.

[22] M. L. Fredman and D. E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, 47(3):424–436, 1993.

[23] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory and Applications*, 5:165–185, 1995.

[24] M. T. Goodrich and R. Tamassia. Dynamic trees and dynamic point location. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 523–533, 1991.

[25] Y. Han and M. Thorup. Integer sorting in $O(n\sqrt{\log\log n})$ expected time and linear space. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 135–144, Vancouver, Canada, November 2002.

[26] J. Iacono. Expected asymptotically optimal planar point location. *Computational Geometry*, 29(1):19–22, 2004. See also SODA'01.

[27] J. Iacono and S. Langerman. Dynamic point location in fat hyperrectangles with integer coordinates. In *Proc. 12th Canadian Conference on Computational Geometry (CCCG)*, 2000.

[28] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983.

[29] R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980. See also FOCS'77.

[30] C. W. Mortensen, R. Pagh, and M. Pătraşcu. On dynamic range reporting in one dimension. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 104–111, 2005.

[31] K. Mulmuley. A fast planar partition algorithm. *Journal of Symbolic Computation*, 10(3/4):253–280, 1990. See also FOCS'88.

[32] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2):166–204, 1981.

[33] M. Pătraşcu. Planar point location in sublogarithmic time. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 325–332, Berkeley, California, October 2006.

[34] M. Pătraşcu. Lower bounds for 2-dimensional range counting. Submitted to STOC, 2007.

[35] M. Pătraşcu and E. D. Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM Journal on Computing*, 35(4):932–963, 2006.

[36] M. Pătraşcu and M. Thorup. Higher lower bounds for near-neighbor and further rich problems. In *Proc. 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 646–654, 2006.

[37] M. Pătraşcu and M. Thorup. Time-space trade-offs for predecessor search. In *Proc. 38th ACM Symposium on Theory of Computing (STOC)*, pages 232–240, 2006.

[38] M. Pătraşcu and M. Thorup. Randomization does not help searching predecessors. In *Proc. 18th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 555–564, 2007.

[39] N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Communications of the ACM*, 29(7):669–679, 1986.

[40] R. Seidel and U. Adamy. On the exact worst case query complexity of planar point location. *Journal of Algorithms*, 37(1):189–217, 2000. See also SODA'98.

[41] J. Snoeyink. Point location. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry (2nd edition)*, pages 767–785. Chapman & Hall/CRC, 2004.

[42] M. Thorup. Approximate distance oracles with polygonal obstacles. Manuscript, 2006.

[43] A. M. Turing. On computable numbers with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, 42:230–265, August 1936.

[44] P. van Emde Boas, R. Kaas, and E. Zijlstra. Design and implementation of an efficient priority queue. *Mathematical Systems Theory*, 10:99–127, 1977. Announced by van Emde Boas alone at FOCS'75.

[45] D. E. Willard. Applications of the fusion tree method to computational geometry and searching. In *Proceedings of the 3rd Annual ACM-SIAM symposium on Discrete Algorithms*, pages 286–295, Orlando, Florida, United States, 1992.