

Statement of Research

Seshadhri Comandur

With the ubiquity of massive data sets, our notions of “efficient algorithms” must change. Reading the whole input may actually be a luxury and we are forced into the *sublinear* realm where decisions need to be made based on a miniscule fraction of the input. It seems necessary to study sublinear time algorithms in a strong theoretical and mathematical framework. I have studied both combinatorial and geometric problems from this perspective and developed tools and techniques to this end. Specifically, I have dealt with *online reconstruction* problems - suppose one is given a large data set with some structural property (sorted numbers, points in convex position, graph that is a tree) that is lost due to noise. We would like to minimally modify the noisy data set to make it have the property again. The catch is that this needs to be done in an online manner. Different sections of the data set should be modified as and when required in *sublinear time*, and we should not go through the whole data set to compute these changes. At some level, it is actually surprising that this can be done.

Data in real life is rarely arbitrary. Usually, inputs to an algorithm come from some distribution - albeit an unknown, probably unrepresentable one. Still, it is natural to expect that there might be some hidden structure in the distribution that might allow for faster algorithms. There is a learning aspect to this problem, as we have to learn some (but not all) features of the highly complex input distribution. Then, we need to use this information to optimize our algorithm. Are such algorithms possible? This was the focus of another part of my research, where we defined a model for this scenario and designed *self-improving algorithms* for a variety of problems.

1 Online Monotonicity Reconstruction

The problem of *monotonicity* has been well studied under the property testing framework [8, 9, 13, 21]. Consider the following problem - there is a function f on the set of integers $[n]$ (or more generally, $[n]^d$, for constant integral d) which we wish to make monotonically increasing. Let $0 < \varepsilon_f < 1$ be the *edit distance* to monotonicity. This is the minimum number of function values that need to be changed to make f monotone, divided by the size of the domain. In joint work with Nir Ailon, Bernard Chazelle, and Ding Liu [1, 2], we designed an algorithm that provides a monotone function g whose distance from f is at most $O(\varepsilon_f)$. The amazing feature of this algorithm is that it works online in sublinear time. We assume that queries come in some sequence and the algorithm is allowed to remember answers to previous queries. Given a query for domain point i , $g(i)$ is computed in polylogarithmic time. There is absolutely no preprocessing and the sequence of queries can be completely adversarial. Tools developed to study functions from the perspective of monotonicity were effectively used by others for streaming algorithms as well [16].

One of the drawbacks of the previous algorithm is that eventually, it takes linear space to remember all previous answers. Also, the queries for g must be handled sequentially and the output function g depends on the sequence of queries. In joint work with Michael Saks [24], we give a construction that only uses sublinear space throughout all the queries, and does not need to remember answers to previous queries. This is not a direct extension of the previous work, and

requires a new set of techniques and insight into monotone functions over lattices. This algorithm can be used for the remarkable task of *parallel reconstruction*. The queries for the function values g can be handled in parallel and all of them can be answered consistently. Among the techniques created was the use of a special data structure over the lattice $[1, n]^d$. There are many exciting directions for further research - it would be worthwhile to investigate how these new techniques and data structures can be used for streaming and sublinear time algorithms. We are looking at reconstruction algorithms for monotone functions on arbitrary posets. It is an interesting research direction to study reconstruction for classes of functions other than monotone ones.

2 Online Geometric Reconstruction

In joint work with Bernard Chazelle [5], we focused on the problem of convexity reconstruction. Given a 3-dimensional surface D , we wish to make it convex in the same online manner discussed above. We output a convex surface D' such that the combinatorial difference between D' and D is (up to constant factors) the minimum possible. We design an algorithm that computes the coordinates of any edge or face of D' in sublinear time. It is quite surprising that this is even possible, since each query looks at a tiny fraction of the surface, yet all the outputs to the queries form a convex surface very close to the input. We give an optimal algorithm for 2-dimensional convexity reconstruction, with almost matching upper and lower bounds. We also provide lower bounds showing a fundamental gap between 2-dimensional and 3-dimensional reconstruction. We use a plethora of tools in our result - halfspace range searching, sampling-based sublinear algorithms for vertex cover, planar separators in sublinear time, and sampling from range spaces of unbounded VC-dimension. There were some very interesting questions (of independent interest) dealing with planar separators that arose from our study, which are part of our current research. It would also be very interesting to design online reconstruction algorithms for other geometric structures.

3 Self-Improving Algorithms

Suppose the inputs to say, a sorting algorithm, are coming from some unknown distribution of permutations \mathcal{D} . In each round, the sorting algorithm is presented with an input generated from \mathcal{D} . The distribution \mathcal{D} is probably highly complex and lacks any closed form representation. If the distribution \mathcal{D} has low entropy, then we would like our sorter to “beat” the standard $\Omega(n \log n)$ lower bound. Fredman [11] designed an algorithm that took exponential preprocessing time on \mathcal{D} and exponential storage, but could then sort an input from \mathcal{D} *optimally*. By *optimally*, we mean that the expected running time is the entropy of \mathcal{D} , which is the information-theoretic lower bound for sorting an input from \mathcal{D} . In joint work with Nir Ailon, Bernard Chazelle, and Ding Liu [3], we design a *self-improving sorter*. Initially, this has a running time guarantee of $O(n \log n)$, and behaves like Quicksort. As it starts sorting permutations from \mathcal{D} , it starts to learn about \mathcal{D} and builds data structures (of near-linear size) storing this information. Quite rapidly, it starts using this information, tuning itself to become faster, and sorts permutations from \mathcal{D} optimally (in the information-theoretic sense explained above). Our self-improving sorter needs to learn very little about \mathcal{D} , and works for a very general class of distributions. We prove a lower bound showing that a self-improving sorter for *all* distributions \mathcal{D} requires exponential space. Furthermore, we prove space lower bounds showing that our algorithm is also space-optimal.

In work with Kenneth Clarkson [6], we take these ideas to the geometric realm. We want to compute Delaunay triangulations in the same framework, where we assume that our input (which is a set of points) is generated from some fixed distribution \mathcal{D} . We use a variety of geometric techniques - ϵ -nets for disks, randomized incremental constructions, optimal expected-case planar search

structures, and linear time Delaunay splitting algorithms. In addition, we resort to information-theoretic tools to prove tight bounds for the running time. The running time of our algorithm matches (up to constant factors) the lower bound for computing the Delaunay triangulation of an input from \mathcal{D} , proving the optimality of our algorithm. One direction for research is to extend this algorithm for higher dimensions, which will probably require the construction of many interesting tools. Currently, we are involved in designing self-improving algorithms for convex hulls. There seems to be a lot of scope for study of many other geometric algorithms in this framework.

4 Sublinear Techniques in Other Areas

A very promising direction of research I have been involved in has been the use of streaming and sublinear techniques for *online learning* algorithms. The online learning problem goes through T rounds. In each round, we choose a point from some convex set in Euclidean space, and then a convex loss function on the set is presented. The loss in a round is simply the function evaluated at our chosen point, and the total loss over T rounds is simply the sum of losses incurred. The loss functions can be adversarial and totally unrelated to each other. A learning algorithm is one that chooses a point in every round, based on the functions that it has seen so far. The standard performance measure is *regret*, which is the difference between the total loss and loss of the best *fixed* decision in hindsight. The aim of learning algorithms is to get $o(T)$ regret, and this has been the study of a lot of past work [17, 19, 25]. In recent work with Elad Hazan [18], we investigated a stronger notion of performance called *adaptive regret*, which is a much better measure of how well the learning algorithm adapts to the loss functions. Noting that present algorithms do not perform well under this measure for most scenarios, we design low adaptive regret algorithms. One of the novel features of this work is the crucial use of sublinear techniques to develop efficient learning algorithms that achieve almost optimal adaptive regret. This result is applicable for a wide variety of problems - portfolio management, online shortest paths, and the tree update problem. We feel that there is a great possibility of further work, and we are currently investigating more applications of our technique to other learning problems, such as learning with switching costs.

Property testing is a well established field with a wide variety of techniques and results [4, 10, 12, 14, 22, 23]. Having an interest in sublinearity, it is natural that I have dealt with algorithms for property testing. Goldreich and Ron [15] first posed the problem of testing expansion in bounded degree graphs. They proposed an algorithm, and stated a very interesting conjecture about the behaviour of random walks in graphs. Resolution of this conjecture would lead to a property tester for expansion. Czumaj and Sohler [7] made some progress in this direction. In independent work, Satyen Kale and I [20] finally designed a property tester with almost optimal parameters, proving interesting statements about random walks. There are very exciting connections to reconstructing expanders and algorithms for graph partitioning that we are investigating.

References

- [1] Ailon, N., Chazelle, B., Comandur, S., Liu, D. *Estimating the Distance to a Monotone Function*, Proc. 8th RANDOM, 2004, 229–236.
- [2] Ailon, N., Chazelle, B., Comandur, S., Liu, D. *Property Preserving Data Reconstruction*, Proc. 15th ISAAC (2004), 16–27.
- [3] Ailon, N., Chazelle, B., Comandur, S., Liu, D. *Self-Improving Algorithms*, Proc. 17th SODA (2006), 261–270.

- [4] Blum, M., Luby, M., Rubinfeld, R. *Self-testing/Correcting with Applications to Numerical Problems*, Proc. 22nd STOC, 1990, 73–83.
- [5] Chazelle, B., Seshadhri, C. *Online Geometric Reconstruction*, Proc. 22nd SOCG, 2006, 386–394.
- [6] Clarkson, K., Seshadhri, C. *Self-Improving Delaunay Triangulations* Submitted to 24th SOCG, 2008.
- [7] Czumaj, A., Sohler, C. *Testing Expansion in Bounded-Degree Graphs*, Proc. 48th FOCS, 2007, 570–578
- [8] Dodis, Y., Goldreich, O., Lehman, E., Raskhodnikova, S., Ron, D., Samorodnitsky, A., *Improved Testing Algorithms for Monotonicity* Proc. 2nd Random, 1999, 97–108.
- [9] Ergun, F., Kannan, S., Kumar, S. Ravi, Rubinfeld, R., Viswanathan, M. *Spot-checkers*, Proc. 30th STOC, 1998, 259–268.
- [10] Fischer, E. *The Art of Uninformed Decisions: A Primer to Property Testing*, Bulletin of EATCS (75), 2001, 97–126.
- [11] Fredman, M. L. *How Good is the Information Theory Bound for Sorting?*, Theoretical Computer Science (1976), 355–361.
- [12] Goldreich, O., *Combinatorial Property Testing - A Survey*, Randomization Methods in Algorithm Design, 1998, 45–60.
- [13] Goldreich, O., Goldwasser, S., Lehman E., Ron, D., Samorodnitsky, A., *Testing Monotonicity*, Combinatorica (20), 2000, 301–337.
- [14] Goldreich, O., Goldwasser, S., Ron, D. *Property Testing and its Connection to Learning and Approximation*, Journal of the ACM (45), 1998, 653–750.
- [15] Goldreich, O., Ron, D. *On Testing Expansion in Bounded-Degree Graphs*, ECCC TR00-020.
- [16] Gopalan, P., Jayram, T. S., Krauthgamer, R., Kumar, R. Estimating the Sortedness of a Data Stream, Proc. 18th SODA, 2007, 318–327.
- [17] Hazan, E., Kalai, A., Kale, S., Agarwal, A. *Logarithmic Regret Algorithms for Online Convex Optimization* Proc. 19th COLT
- [18] Hazan, E., Seshadhri, C. *Testing Expansion in Bounded-Degree Graphs* ECCC TR07-076 (Submitted to 40th STOC, 2008)
- [19] Kalai, A., Vempala, S. *Efficient Algorithms for Online Decision Problems* Proc. 16th COLT, 2003.
- [20] Kale, S., Seshadhri, C. *Testing Expansion in Bounded Degree Graphs*, ECCC TR07-076.
- [21] Parnas, M., Ron, D., Rubinfeld, R. *Tolerant Property Testing and Distance Approximation* Journal of Computer and System Sciences (72), 2006, 1012–1042.
- [22] Ron, D. *Property Testing*, Handbook on Randomization, Volume II, 2001, 597-649.
- [23] Rubinfeld, R., Sudan, M. *Robust Characterization of Polynomials with Applications to Program testing*, SIAM Journal of Computing (25), 1996, 647–668.
- [24] Saks, M., Seshadhri, C. *Parallel Monotonicity Reconstruction* Proc. 19th SODA, 2008.
- [25] Zinkevich, M. *Online Convex Programming and Generalized Infinitesimal Gradient Ascent* Proc. 20th ICML, 928–936.