

Statement of Research

C. Seshadhri

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 a wide variety of problems from this perspective and developed tools and techniques to this end. A lot of my recent work has dealt with the use of random walks to speed up computation on graphs. A random walk is a very simple algorithmic procedure: we start from a vertex and repeatedly move to a random neighbor for some prescribed number of steps, called the *length*. When the starting vertices and number and length of such walks is chosen carefully, we can gain a surprisingly large amount of information about the graph. This tool can be used very effectively to learn many properties of a graph in sublinear time. In addition to yielding such algorithms, our results also give many combinatorial theorems of independent interest. We focus mainly on sparse graphs¹ since this is where this technique is most powerful. Also, for sparse graphs, not much is known about what can be achieved in sublinear time.

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 a model was defined for this scenario and *self-improving algorithms* were designed for a variety of problems.

1 Testing and Reconstructing Expanders

Property testing is a well established field with a wide variety of techniques and results [BLR90, Fis01, Gol98, GGR98, Ron01, RS96]. Given a property \mathcal{P} , we have a relaxed version of the standard decision problem: accept an input I if $I \in \mathcal{P}$, and reject I if it is *far* from \mathcal{P} . Suppose our domain is graphs. Then an input G is far from \mathcal{P} if G needs to be changed in many edges to have \mathcal{P} . For a large variety of properties, one can find sublinear (or even constant) time algorithms for this relaxed problem. Testing for properties of sparse graphs has been a very exciting area of research [GR02, GR99] with lots of many open problems. One very natural property to study is that of being an *expander*. Goldreich and Ron [GR00] first posed this problem, and an important connection to property testing in general was shown [CS07a]. After no progress for many years, Czumaj and Sohler [CS07b] made the first progress in this direction. In independent work with Satyen Kale [KS08], we resolved this conjecture and designed a property tester with optimal parameters.

This work led to the question of *reconstructing* expanders. Suppose one is given a graph G that is not an expander. Can we, in sublinear time per edge, find out which edges to add to G , so that it becomes an expander? In other words, can we quickly decide which edges to add to

¹These are graphs where the number of edges is linear in the number of vertices.

boost the connectivity of G ? This led to the investigation of random walks on graphs that are very close to being expanders. Suppose one has an expander to which a very small non-expanding component is added. Will walks within the expander component still have some mixing properties? Previous techniques were unable to give any results for such scenarios. With Satyen Kale and Yuval Peres [KPS08], we prove some very interesting theorems about such random walks. We also develop techniques to distinguish vertices in the large expanding component from the remaining vertices in sublinear time. There appears to be many possible directions to carry forward this work. Graph partitioning is a very well studied problem, where one desires to break up a graph into small very well connected components. Our techniques may provide efficient algorithms for such partitioning problems. Recently, there has been a lot of interest in *local partitioning algorithms*, which very efficiently find such well connected components. We strongly believe that the approaches we have used will be helpful for a better understanding of this problem.

Some parameters of our reconstruction algorithm are not optimal, and this seems to be intimately connected to the non-optimality of many graph partitioning algorithms. It would be very interesting to see if this is inherent to these approaches. If we can break this barrier, then that would imply very strong theorems about graph partitioning.

2 Finding Cycles and Trees in Graphs

Many interesting graph properties can be expressed as the non-existence of some substructure. For example, forests are graphs that contain no cycles. An interesting class of such properties is *minor-freeness*. A graph H is a minor of G , if H can be obtained from G through edge or vertex deletions, and edge contractions. A graph G is H -minor free if it contains no H -minor. The classical Kuratowski-Wagner theorem states that planar graphs are exactly those that are K_5 or $K_{3,3}$ minor-free². Forests are K_3 minor-free graphs. These are the *forbidden* minors for these properties. Many properties can be expressed through minor freeness. In some of the most fundamental work in graph theory, Robertson and Seymour [RS95, RS04] have shown many deep theorems about such properties.

Given a graph that is far from having such a “freeness” property, can we find a violating sub-structure in sublinear time? This question was first formalized in [BSS08], although related questions have been studied for almost a decade. The first progress towards this was the focus of joint work with Oded Goldreich, Dana Ron, and Asaf Shapira [GRSS]. We provide an optimal sublinear time algorithm that finds cycles in graphs that are far from being forests. This algorithm uses random walks to find such cycles and uses tools from [GR00] about connections between random walks and cuts. We extend our algorithm to find cycles of any fixed length and give a completely different (constant time) algorithm to find tree minors. All our results put together lead to a greater understanding of the hardness of finding forbidden minors, and why it is closely connected to finding cycles.

This work is just the tip of the iceberg of minor freeness algorithms. The very natural question of finding forbidden minors in planar graphs is very much open, and will probably require new techniques. A long term project would be to get a sublinear time algorithm to find H -minors in a graph that is far from being H -minor free, for any fixed graph H . This would require a better understanding of the work of Robertson and Seymour, and will lead to very interesting combinatorial results.

²The graph K_r is the complete graph on k vertices. The complete bipartite graph between sets of r and s vertices is denoted by $K_{r,s}$.

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 small 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 [Fre76] 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 [ACCL06], 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.

This was then taken to the geometric realm. In work with Kenneth Clarkson [CS08], we 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 wide 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. With Kenneth Clarkson and Wolfgang Mulzer [CMS10], we design a self-improving algorithm for convex hulls. This requires a completely different perspective to this classical problem. We show connections to output sensitive algorithms, whose running time depends on the output size (which can be much smaller than the input). We also gain a deeper understanding of convex hulls of independently distributed points. There seems to be a lot of scope for further work. Giving a self-improving algorithm for Voronoi diagrams and convex hulls in higher dimensions would be of great interest. Such algorithms might be able to have some kind of output sensitivity, something that has not yet been achieved for these problems.

4 Sublinear Techniques in Online Learning

I have been involved in 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 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 [HKKA06, KV05, Zin03]. In work with Elad Hazan [HS09], 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 are currently investigating more applications of our technique to other learning problems, such as learning with switching costs.

References

- [ACCL06] N. Ailon, B. Chazelle, S. Comandur, and D. Liu. Self-improving algorithms. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 261–270, New York, NY, USA, 2006. ACM.
- [BLR90] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing (STOC)*, pages 73–83, 1990.
- [BSS08] I. Benjamini, O. Schramm, and A. Shapira. Every minor-closed property of sparse graphs is testable. In *Proceedings of the Fourtieth Annual ACM Symposium on the Theory of Computing*, pages 393–402, 2008.
- [CMS10] K. Clarkson, W. Mulzer, and C. Seshadhri. Self-improving algorithms for convex hulls. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010.
- [CS07a] A. Czumaj and C. Sohler. On testable properties in bounded degree graphs. In *Proceedings of the 18th Annual Symposium on Discrete Algorithms (SODA)*, pages 494–501, 2007.
- [CS07b] A. Czumaj and C. Sohler. Testing expansion in bounded degree graphs. In *Proceedings of the Annual 48th Symposium on Foundations of Computer Science (FOCS)*, pages 570–578, 2007.
- [CS08] K. L. Clarkson and C. Seshadhri. Self-improving algorithms for delaunay triangulations. In *Proceedings of the 24th Annual Symposium on Computational Geometry (SOCG)*, pages 148–155, 2008.
- [Fis01] E. Fischer. The art of uninformed decisions: A primer to property testing. *Bulletin of EATCS*, 75:97–126, 2001.
- [Fre76] M. Fredman. How good is the information theory bound in sorting? *Theoret. Comput. Sci.*, 1(4):355–361, 1975/76.
- [GGR98] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.
- [Gol98] O. Goldreich. Combinatorial property testing - a survey. *Randomization Methods in Algorithm Design*, pages 45–60, 1998.
- [GR99] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.
- [GR00] O. Goldreich and D. Ron. On testing expansion in bounded-degree graphs. Technical report, ECCO, 2000.

- [GR02] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.
- [GRSS] O. Goldreich, D. Ron, A. Shapira, and C. Seshadhri. Finding cycles and trees in sublinear time. Manuscript.
- [HKKA06] E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *COLT*, pages 499–513, 2006.
- [HS09] E. Hazan and C. Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, page 50, 2009.
- [KPS08] S. Kale, Y. Peres, and C. Seshadhri. Noise tolerance of expanders and sublinear expander reconstruction. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 719–728, 2008.
- [KS08] S. Kale and C. Seshadhri. Testing expansion in bounded degree graphs. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP(I))*, 2008.
- [KV05] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.
- [Ron01] D. Ron. Property testing. *Handbook on Randomization*, II:597–649, 2001.
- [RS95] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory Series B*, 63(1):65–110, 1995.
- [RS96] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal of Computing*, 25:647–668, 1996.
- [RS04] N. Robertson and P. D. Seymour. Graph minors. XX. Wagner’s conjecture. *Journal of Combinatorial Theory Series B*, 92(1):325–357, 2004.
- [Zin03] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference (ICML)*, pages 928–936, 2003.