Jeff Kinne
Research Statement

My research interests are within theoretical computer science, specifically complexity theory and the study of randomized computations. Complexity theory aims to determine how efficiently problems can be solved with respect to various resources, such as time and memory space, and how the resources relate to each other. Two very natural and significant questions are the following.

- Is finding proofs for theorems more difficult than verifying their correctness? This is the famous "P versus NP" question and is equivalent to whether any of a host of NP-hard optimization problems – traveling salesperson, graph coloring, maximum satisfiability, etc. – can be solved efficiently.
- Does randomness make algorithms more powerful, or does every randomized algorithm have an efficient deterministic simulation?

Decades of investigation by leading researchers have produced beautiful insights and tools for studying both problems. Still, the final answers to these two fundamental questions remain open. My research has mainly focused on the second question, the power of randomness. The topic is intriguing because many important problems can be solved by randomized algorithms that are either much faster or much simpler to implement than the best-known deterministic algorithms.

The area of *derandomization* aims to convert randomized algorithms into deterministic algorithms without much loss in efficiency. Famous recent contributions include the deterministic time-efficient algorithm for primality testing of Agrawal et al. [1] and memory-space efficient algorithm for connectivity on undirected graphs of Reingold [2], both achieved by derandomizing well-known randomized algorithms. A long line of research into general-purpose methods of derandomization has led to the widely-held *conjecture* that *all* time-efficient randomized algorithms can be converted into time-efficient deterministic algorithms. In particular, such derandomization is implied by a very reasonable complexity-theoretic hardness assumption – that there is a problem that can be solved in time $2^{\Theta(n)}$ but cannot be solved by a circuit that has only $2^{o(n)}$ gates. The hard problem is used to construct a "pseudorandom generator" which outputs a small set of random strings to use in executing the randomized algorithm.

In practice, this approach to derandomization may be too inefficient because the randomized algorithm needs to be executed once for each of a polynomial number of pseudorandom strings. Further, the hardness assumption, though plausible and widely believed in the community, has been notoriously difficult to prove and recent work has shown that in fact *any* non-trivial derandomization of certain randomized algorithms implies the existence of a hard problem.

Typically-Correct Derandomization The difficulties of derandomizing all efficient randomized algorithms suggest that some new direction is needed to make further progress. A promising candidate that has emerged is to look at deterministic simulations that are allowed to make a small number of mistakes, so-called "typically-correct derandomization".

- For standard derandomization, also known as "everywhere-correct" derandomization, we want a deterministic algorithm that is correct on *every* input.
- For *typically-correct derandomization*, the deterministic algorithm only needs to be correct on at least $1 - \delta$ fraction of the inputs, for a small δ.

The hope is that derandomization with a small number of mistakes is (i) useful in some applications, and (ii) possible to achieve *very efficiently* with current techniques.

I along with Dieter van Melkebeek and Ronen Shaltiel [3] demonstrate that indeed typically-correct derandomization is possible for a number of interesting classes of randomized algorithms, including randomized streaming algorithms, constant-depth circuits, and sublinear-time algorithms. Moreover, the derandomization is very efficient. Whereas the approach to everywhere-correct derandomization discussed above executes the randomized algorithm many times, our approach deterministically selects *a single random string* and executes the randomized algorithm *only once*.

The hope is that such typically-correct derandomization can be achieved for *all* efficient randomized algorithms, without the need for a hard problem which, as discussed above, is known to imply everywhere-correct derandomization. We demonstrate a very small δ for which typically-correct derandomization still implies the existence of a hard problem. We expect that derandomization for slightly larger values of δ is possible without the need for a hard problem, and we continue to work in this direction. As a first step, we show that typically-correct derandomization of all randomized algorithms follows from a weaker hardness assumption than the one stated previously that implies everywhere-correct derandomization.

Hierarchy Theorems for Randomized Algorithms Another way to make progress in our understanding of randomized algorithms is to study their fundamental properties. Among the most basic questions is whether an algorithm with more resources can accomplish more than an algorithm with fewer resources.

- *Time and space hierarchy theorems* state that an algorithm with more time or memory space can solve strictly more problems than algorithms with fewer resources.

Such hierarchies were among the first properties proved of deterministic algorithms. However, the proof techniques do not work for randomized algorithms, and the fundamental question of whether time and space hierarchy theorems hold for randomized algorithms remains open. For example, although unlikely and counter-intuitive it may be that all problems solvable by a randomized algorithm in time $n^{\log n}$ can also be solved by a linear-time randomized algorithm.

Recent work by a number of researchers has come the closest yet to solving the longstanding problem by showing *time* hierarchy theorems for *slightly non-uniform* randomized algorithms. Rather than consisting of a single algorithm, a non-uniform algorithm consists of a small set of algorithms where the choice of which to execute is made "non-uniformly" depending on the size of the input. The best results use the smallest non-trivial amount of non-uniformity, by having only two algorithms to choose between for each input length. Dieter van Melkebeek and I [4] demonstrate such hierarchy theorems for *memory space* rather than time. We also derive hierarchy theorems for other types of algorithms, including for example quantum algorithms and Arthur-Merlin protocols.

We hope that we can remove the remaining small amount of non-uniformity from these results and finally settle the major open problem of proving hierarchy theorems for randomized algorithms.

Future Research I continue to work on questions that have been raised by my previous research. In one ongoing project, Dieter van Melkebeek and I [5] are investigating whether all randomized *monotone* computations can be derandomized.

A second project is to explore the implications and uses of typically-correct derandomization throughout complexity theory. One potential application is in "time-space lower bounds" for NP-complete problems such as Boolean formula satisfiability (SAT). The widely-held conjecture is that efficient randomized algorithms cannot compute SAT, but even randomized algorithms that

run in linear time have not yet been ruled out. For deterministic machines there has been some progress. Recent work has shown that deterministic machines that run in time roughly $n^{1.8}$ and use small memory space cannot compute SAT. However, the techniques fail for randomized machines. Further, the general-purpose approach for everywhere-correct derandomization discussed earlier would result in deterministic simulations that take more than $n^{1.8}$ time and so would fall short of establishing time-space lower bounds for randomized algorithms computing SAT. Typically-correct derandomization with strong enough parameters, on the other hand, *would* imply such a result. This connection to time-space lower bounds is thus one area where the very small loss in efficiency of typically-correct derandomization is key.

I am also exploring new research directions within complexity theory. A research project that I have begun and am very excited about attempts to answer whether efficient algorithms can compute NP-complete languages "on the average", i.e., give a correct answer on most inputs. A definitive answer in either direction would have important practical implications. An affirmative answer would be useful because NP-complete problems need to be solved in innumerable situations, and oftentimes solutions that work well "on average" are acceptable. A negative answer would be a major result because it would in particular solve the P versus NP problem. A major step towards understanding whether NP-complete languages are hard on average is to establish a "worst-case to average-case connection" for NP-complete problems, i.e., to show that P $\neq$ NP implies NP-complete languages that are hard on average. The hope is that such a connection can be used to strengthen the foundations on which modern cryptosystems rely. The security of cryptosystems currently relies on *average-case* hardness of NP problems such as factoring and discrete logarithm that are "isolated" in the sense that their computational hardness does not relate to many other problems. The goal is to use a worst-case to average-case connection for NP-complete problems to base cryptosystems on the much weaker and more believable assumption of *worst-case* hardness of NP-complete problems – the most difficult problems in NP, including thousands of problems from diverse fields.

Concluding Remarks Partaking in cutting-edge research and expanding the knowledge of our society are some of the main reasons that being a professor of computer science is my dream job. I have chosen complexity theory as my primary research area because there are so many problems to solve that are very important but at the same time are notoriously difficult to solve and involve deep mathematical insights.

I am drawn to academia, as opposed to industry, because of (i) my desire to teach and (ii) the opportunity to guide and assist students with their research efforts. To accomplish the latter I will use many of the same principles that I believe are key to effective teaching: choosing research projects that are exciting and relevant, setting clear goals, being organized with regular meetings to exchange ideas and report progress, etc. In terms of student research topics, I am glad to bring others on board my research projects, provide research ideas tuned to the background of the student, or expand my research horizons to whatever the students' interests are – inside or outside of theoretical computer science. This also applies to potential collaboration with colleagues in other research areas. I am particularly interested in collaborating and providing "the theory perspective" in areas such as multi-core/parallel algorithms, cloud computing/SaaS, security, and computational science or other areas that will play an increasing role in technology and science in the coming years.

At UW-Madison I have developed a passion for research, and I look forward to continuing my exploration of the computational landscape in the future.

References

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxen. Primes is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[2] Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4), 2008.

[3] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators, typically-correct derandomization, and circuit lower bounds. Invited to appear in the special issue of *Computational Complexity* for selected papers from *Proceedings of the 13th International Workshop on Randomization and Computation* (RANDOM), 33 pages. Preliminary version appeared in RANDOM, LNCS 5687, pages 574–587, 2009.

[4] Jeff Kinne and Dieter van Melkebeek. Space hierarchy results for randomized and other semantic models. *Computational Complexity*, 53 pages, 2009. In press. Preliminary version appeared in *Proceedings of the 25th International Symposium on Theoretical Aspects of Computer Science* (STACS), pages 433–444, 2008.

[5] Jeff Kinne and Dieter van Melkebeek. On derandomization and average-case complexity of monotone functions. Working paper, 15 pages, 2009.