Teaching Statement

Laxman Dhulipala

One of the most impactful aspects of being in academia is teaching courses and mentoring students. I am very excited to help guide, teach, and work with the next generation of computer science students.

Teaching Experience

I find teaching to be a highly rewarding activity, and have helped teach, hold office hours and labs, and set problems for homeworks and exams as a teaching assistant for 7 courses throughout my undergraduate and graduate career. As a graduate teaching assistant for the Undergraduate Complexity Theory course, I noticed that students at my office hours often had similar conceptual misunderstandings, and so I would devote time to work step-by-step through key concepts and examples with them. During one weekend, I spent several hours with students working on a space-complexity problem set to help them understand the equivalence of two characterizations of the complexity class NL (Nondeterministic Log-space). Our discussion covered many topics, including the basics of space complexity, how objects can be encoded as inputs to a Turing machine, and carefully worked examples of space-efficient reductions. Reviewing this material step-by-step helped them understand *the key abstractions and methods used to design space-efficient algorithms*, and I was thrilled to see that they were able to solve the remaining problems on the problem set without issues.

I have also had the opportunity to get hands-on teaching experience by giving guest lectures, and preparing lecture notes in courses at CMU. As a lead teaching assistant for the Algorithms in the Real World course, I helped write *An Introduction to Parallel Algorithms*, a set of lecture notes covering the work-depth model, and a variety of parallel algorithms and algorithm design techniques. I also gave two guest lectures on parallel algorithms for this course. One lecture was on graph compression, and covered theoretical topics such as hardness results for graph reordering problems, as well the practical algorithms that are used to solve these problems in the real-world. I also gave a lecture on graph connectivity, covering classic approaches, as well as a recent work-efficient algorithm from my research based on low-diameter decompositions. The students in the course gave very positive feedback about these lectures, and many students from the class chose to work on parallel algorithms for their final projects, seeking me out to provide feedback on their ideas.

Teaching Approach

I am very interested in training students in how to conceptually model and approach problems. I believe that hidden underneath complex algorithms and proofs are elegant and powerful ideas that everyone can understand. My goal when teaching is to help everyone understand the core ideas, and to gradually add details until the full solution is "obvious". Since everyone has different strengths and approaches when learning, it is beneficial to present the same concepts from many perspectives and angles, while keeping the content varied and exciting. Two ways that I will help students gain practice with modeling and approaching problems are:

(1) **Providing hands-on experience.** *I am very interested in designing hands-on exercises that have students analyze and implement parallel algorithms.* Exercises could range from simple problems such as prefix-sums to more complex parallel algorithms for sorting, dynamic programming, or even graph algorithms, which are built by combining basic parallel primitives. One of the key pedagogical challenges is *providing students with tools for parallel programming that are easy to use and reason about*. I believe that *the libraries and systems developed throughout the course of my research are ideal platforms for teaching parallel programming*. For example, I have developed the ParlayLib library,[1] an open-source parallel programming library that provides a broad set of useful parallel primitives such as sorting, selection, and many others. I will develop assignments that have students design and implement parallel algorithms using the high-level primitives from ParlayLib, reason about the theoretical costs of their implementations, and evaluate the practical performance and speedup of their implementations on multicore machines. I am excited to pursue a similar approach when teaching parallel graph algorithms using the frameworks and systems designed in my research, such as GBBS and Aspen.[2]

[1] https://github.com/cmuparlay/parlaylib

[2] https://github.com/ParAlg/gbbs and https://github.com/ldhulipala/aspen

(2) **Integrating research topics into the curriculum.** I believe that *incorporating research topics and challenges into both lectures and homework makes students more motivated and excited about the subject*. As an example, in the Algorithms in the Real World course, I designed a multi-part problem that had students explore different parallel algorithms for computing shortest paths, and in the final part analyze a parallel algorithm to construct hopsets, a widely used tool in parallel and dynamic algorithms. Similarly, weaving illustrations of challenges still waiting to be solved into lectures, and presenting the practical implications of course material can help students understand the relevance of what they are learning. I am very interested in designing course content that has students *creatively apply and understand the material*, and believe that this approach will help make the content more meaningful, memorable, and help encourage students to seek out research opportunities and pursue other creative aspects of computer science in the future.

Advising Approach

I am very interested in creating a collaborative and friendly environment in my research group which cultivates curiosity and encourages creativity. I plan to adopt an advising style which matches the maturity of the students, becoming more hands-off as the student becomes more comfortable leading the direction of the research. For example, in my previous experiences advising undergraduates at CMU and working with Ph.D. students at MIT, my interactions have ranged from working through low-level performance aspects of a piece of code together, to providing advice about how to present a technical concept in a paper. I believe that holding weekly meetings with students is a very effective way of keeping research on track, while giving students an opportunity to reflect on their ideas and make progress on their own.

I also plan to host a weekly group lunch and a reading group on parallel algorithms. I have organized a reading group on parallel algorithms for several semesters at CMU, and am currently organizing a similar group at MIT. I have personally found both group lunches and reading groups to be valuable opportunities to build a sense of community within a research group.

Example Courses

I am excited to teach a number of new courses which I am interested in developing, as well as helping to teach existing undergraduate algorithms courses, which are both listed below:

Undergraduate.

(1) **Foundations of Parallel Computing.** A course introducing students to *parallel thinking*—how to think about algorithms that perform multiple operations at the same time—through the design and analysis of parallel algorithms. It will give students experience writing and reasoning about parallel programs.

(2) **Algorithm Engineering.** An upper-division, research-focused course investigating the principles of designing and implementing efficient algorithms in different models of computation, e.g., external and semi-external memory algorithms, streaming algorithms, and parallel algorithms. The course would involve reading research papers and a final project.

Graduate.

(1) **Efficient Graph Processing Systems.** A research seminar on recent advances in graph processing systems, including practical parallel, concurrent, and dynamic graph algorithms. Students will be exposed to current research topics in graph processing from both theoretical and practical perspectives.

(2) **Advanced Parallel Algorithms.** A course covering fundamental parallel models (shared-memory/PRAM, Massively Parallel Computation, and others), techniques for designing efficient parallel algorithms, and recent research results and challenges in this area. The course will include a term-long project.