Teaching Statement
Chang Liu

My goal in teaching is to provide students with the ability to pursue computer science research and practices. Specifically, I would like to educate students with rigorous thinking ability and solid implementation skills, with an emphasis on providing students with hands-on experiences to solve real problems. My teaching experiences make me realize that not only my students, but also myself could benefit from my teaching.

Teaching experience. I worked as a teaching assistant for three courses: Special Topics in Deep Learning with Prof. Dawn Song, Introduction to Compilers (CMSC 430) with Prof. Jeffery Foster, and Parallel Algorithms (ENEE 699) with Prof. Uzi Vishkin. Introduction to Compilers is a senior-level undergraduate course that covers basic concepts and techniques to build a compiler. As a teaching assistant, I gave lectures and hosted office hours to help students to understand the organization of a typical compiler via implementing different parts of a toy example.

Parallel Algorithms was a graduate-level theory-oriented course. I helped students to understand the challenges when converting a sequential algorithm into its parallel counterpart, and the techniques to tackle them. By communicating with students, I deepened my understanding of designing parallel algorithms as well. This experience finally contributed to my work on Oblivious Network RAM published in Asiacrypt 2015, that discussed solutions to the parallel Oblivious RAM problem.

Special Topics in Deep Learning is a graduate research course. I helped students to form their research problems, and guided them to finish a semester-long project.

Before coming to US, I had six years of part-time tutoring experiences in China. My students were from elementary schools to high schools. I taught them the skills of algorithm design and programming to prepare for National Olympiad in Informatics in Provinces (NOIP), which is the largest high-school regional programming competition in China. These tutoring experiences not only equipped my students with programming skills, but also let me learn how to explain abstract algorithms to students with little background knowledge via intuitive examples. Further, my students’ achievements were my great gift that made me feel fulfilled. I could still remember how I was impressed and proud when a smart elementary school student of mine implemented a dynamic programming algorithm to solve a simplified knapsack problem under my help. After years, my first tutored student contacted me and told me that he will be interning at Facebook. He appreciated my work for providing him skills to pass the job interview.

Teaching interests. With my research experiences in security, programming languages/compilers, and distributed systems, I am qualified and interested in teaching courses on related topics. I would like to design courses to allow students to practice with real problems. Specifically, I am mostly interested in teaching deep learning courses, either at the undergraduate level or at the graduate level. I would like to combine my own research experience into the course materials to allow students to have a better understanding of both traditional approaches, and the state-of-the-art.

In system security related courses, I would prepare materials for students to practice attacks and defenses. In programming languages, I would enjoy teaching courses on compiler design and implementation, program analysis, and program verification and synthesis, and I would develop assignments to guide students to implement compilers to meet certain design goals. In distributed systems, I am interested in courses that
cover topics on using large-scale distributed computation systems such as Hadoop or Spark to implement data analytical algorithms. I would particularly enjoy teaching courses at the intersection of my research areas.

Besides traditional courses, I am interested in teaching competitive programming courses, which have not been provided in most US universities. Programming competitions, such as ACM International Collegiate Programming Contest (ICPC) and TopCoder, require participants to design and implement efficient algorithms to solve certain problems within a time constraint. A competitive programming course will benefit both research-oriented and engineering-oriented students. On one hand, since competitive programming examines a programmer’s comprehensive ability to solve problems, its variants called code interview are commonly adopted in the job interview process of IT companies as Google and Facebook. Students seeking for an industrial job usually spent extracurricular time to exercise related skills for code interview. Therefore, providing such a course would benefit students pursuing an industrial career. On the other hand, my personal experiences on ICPC during my undergraduate study provide me with a solid foundation to implement my PhD research projects. This observation evidences that competitive programming may be helpful for students pursuing a research career as well. My experiences in ICPC also qualify me to teach such a courses.

Mentoring. I enjoy mentoring students, and I realize that this activity may eventually benefit my own research.

In Berkeley, I have been working with undergraduate students such as Xinyun Chen and Xiaojun Xu. We have finished several research projects, which eventually lead to publications in top conferences. Xinyun Chen is now a PhD student at UC Berkeley, and Xiaojun Xu is applying this year.

During the ObliVM project, I collaborated with many students. I mentored an undergraduate students, Kevin Sekniqi who is now a PhD student in Cornell, to improve the compiler. His work was finally incorporated into the open source release. I taught Xiao Wang, a PhD student at University of Maryland, to program in the ObliVM language, while he provided me valuable feedback on language syntax and features. These communications greatly influenced the design of ObliVM to make it user-friendly.

I co-hosted the ObliVM+Obliv-C week with Prof. Elaine Shi at Cornell. During the week, I presented a tutorial on how to program in ObliVM, and asked attendees who had little programming language background to practice programming using the ObliVM language. The feedback from attendees provided me deeper understandings of the burden that a non-expert programmer might be facing, as well as a guidance for future improvements. Now one master student from Cornell, Vivek Gaddam, is working with me on developing oblivious data structure libraries. These efforts will contribute to the open sourced code base.