Teaching Statement

Xin Zhang

Objectives. As an educator in computer science, I strive to:

1. *Actively engage students.* Both in my experiences as a learner and an educator, I find students most productive when they are actively engaged. I still remember the first time I gave a lecture – it was at 6.820: Foundations of Program Analysis, a graduate-level course in computer science at MIT. I was eager to do it well so I spent a lot of time rehearsing. The result was that while I presented the materials fluently, I found students gradually lost attention. That experience made me realize that classroom teaching is very different from academic presentation, where one's goal is to describe one's work in a short time to a group of experts. Similarly, when I was a student, I struggled the most in lectures that were mostly one-sided presentations from the instructor. After each lecture, I could only remember what points had been covered and had to study them myself afterwards to really understand. A common pitfall of teachers in these stories is to treat teaching as content delivery. Instead, teaching should be a process where teachers facilitate the students to actively explore the unknown. Thus, being able to engage students is crucial to be a successful educator. Needless to say, compared to classroom teaching, it is even more important for mentoring students in research.

2. *Help students understand philosophies behind techniques.* Computer science is a field that changes rapidly, and to stay relevant in it one has to keep studying through one's lifetime. The key to this seemingly endless process is that, while new techniques keep coming out, the high-level paradigms behind them often vary little from existing ones. For example, while the syntax of C++ and Java are different, they are very similar in many designs. As a result, it often takes little time for one to pick up one of them after mastering the other. As a teacher, I want the students to understand the high-level paradigms and the rationale behind certain design choices. Such knowledge can easily transfer to other problems and domains, and I collectively call this knowledge "philosophies". In the context of research, "philosphies" mean the skills to define and approach a research problem.

Methods. To achieve the two above objectives in both classroom teaching and research mentoring, I will adopt several methods that I have learnt from training programs and applied in past experiences.

Classroom Teaching. In any course I teach, I will try to involve heavy hands-on experience and apply active learning techniques, which I elaborate below.

Given computer science is an applied area, what is better to motivate students and force them to engage than ask them to solve real-world problems through hands-on experiences? Students often form a deep understanding of the knowledge through hands-on experiences. I have applied hands-on experience in forms of problem sets such as coding assignments, and semester-long projects, and they indeed helped students understand the course content in many aspects. When I was a teaching assistant for Software Analysis and Testing, a graduate-level course of over 40 students at Georgia Tech, there were several lectures on various bug finding techniques. For these lectures, I designed a problem set where the students were asked to use previously learnt techniques to discover bugs in programs that I collected from GitHub. In the feedback submitted by the students later, most students wrote that these problem sets made them not only understand the techniques better but appreciate their values in practice. As another example, when I was a teaching assistant for an undergraduate-level database class at Georgia Tech, other teaching assistants and I designed a project where students were asked to form groups and design a system for an imaginary car rental company like Zipcar. It was a class of around 200 students from different majors. Similar to the previous example, the students told us in the feedback that, they learnt about the practical value of the course content and formed a deeper understanding of it through the project. Moreover, they also learn about soft skills like how to work in teams and how to present one's work. What was the most interesting to me was that the team receiving the highest score was formed by two students in Industrial Engineering. Although they might not be technically stronger than the computer science students, they worked better as a team.

I have systematically learnt about active learning in a teaching course at Georgia Tech, and a teaching certificate program at MIT. They cover a wide range of active learning methods. For example, in "Think Break", the teacher asks students a rhetorical question, and then allows 20 seconds for them to think about the problem before the teacher goes on to explain. The Think-Break technique encourages students to take part in the problem-solving even when discussion is not feasible. I

will use it often in classes of all sizes. As another example, in "Concept Mapping", students are asked to write keywords onto sticky notes and then organize them into a flow chart. I plan to use it in seminar courses where I divide students into groups to present different concepts that have been covered. Last but not least, it is very important to get timely and accurate feedback from the students so that I can adjust the pace of the courses and reflect on the teaching methods I use accordingly. This can be achieved through quizzes, homework, and in-class active learning techniques such as doing anonymous polls using clickers.

Research Mentoring. I plan to adjust my mentoring style based on the types of students. I have worked with students ranging from master's students, 1st-year Ph.D. students, to Ph.D. students that are close to graduation, and my roles were very different. Although my approaches may vary by students, there are a few best practices I want to follow:

First, I will make students feel they own the research. Motivation is the most important factor to a student's success in research. In my past experiences, if a student has the ownership of a project, they usually have a stronger sense of responsibility and feel a greater sense of achievement when progress is made. On the other hand, if a student is mostly following the advisor's order, they often struggle to meet the expectations and the interaction is stressful for both sides. Moreover, the end goal of Ph.D. programs is to produce independent researchers so the students will need to be able to finish a project independently at some point.

Second, I will be patient with junior students. It is a very large jump from the acquiring-content-style undergraduate learning to doing research. But what is amazing about students is that they often grow "exponentially" as time passes. In the 3rd year of my Ph.D., I worked with a 1st-year Ph.D. student from another area who wanted to apply techniques from my area. I started by spending a lot of time sitting with the student to work on the problem together. But I quickly became frustrated as the student and I did not speak the same research language. I ended up solving the problem mostly by myself. A few weeks later, the students slowly began to understand my solution and the rationale behind it. Much to my surprise, he even found an error in my solution. All my effort in the beginning paid off–it just took some time for the student to get started. Thus, it is important for the advisor to be patient with junior students and invest enough time in the beginning.

Finally, it is equally important to provide moral support to students as to provide technical support. A major difference between academia and industry is that the feedback loop in academia is much longer. It often takes from a few months to a year for a student to submit a paper and multiple tries to get it accepted. Many students I have interacted with had their first submissions rejected and almost all of them showed different degrees of frustration. Why do reviewers not like my work? Is it bad? Has my effort in the past year been wasted? It is important for an advisor to encourage the student when these thoughts appear. In general, the advisor should stay positive and encouraging in front of students. Being a tenure-track assistant professor can be stressful. However, if the advisor appears stressed or negative to student, it can make the student even more unsettled.

Future Courses. I am most interested in teaching courses related to my core research areas at both undergraduate and graduate levels, including programming languages, compilers, formal methods, and software engineering. In addition, at undergraduate level, I am willing to contribute in teaching introductory programming, algorithms, data structures, and database systems. At graduate level, based on my expertise, I can teach advanced courses such as program analysis, probabilistic programming, program synthesis, theorem proving, and constraint solving. Tied to my research directions, I am interested in creating new courses in emerging topics such as trustworthy AI, machine learning augmented formal methods, and logic-probabilistic reasoning.

In summary, I am passionate about both teaching and advising students in research. I believe students are the main subjects in learning. It is my job to create a supportive environment to help them learn and grow.