Kirshanthan Sundararajah
Teaching Statement

I believe that teaching strengthens one's understanding of the subject matter, and in return, it makes them better teachers. Every time I teach a subject that I think I am thorough of gives me a brand new perspective, making teaching enjoyable. In my opinion, teaching is a medium of communication for your understanding of a subject. Therefore, finding effective strategies to communicate understanding helps achieve our teaching goals. I envisage that developing strategies for effective communication of understanding through having many different teaching experiences.

Teaching Experience

I have teaching experience from universities in two different countries, and I have served as a teaching assistant and an instructor. These experiences have shaped my thoughts on both learning and teaching. My different teaching experiences largely contributed to developing a concrete teaching philosophy.

Teaching Assistantships

My very first teaching experience in computer science was being a teaching assistant for **Programming Fundamentals (CS 1032)** class at the University of Moratuwa. I was mainly responsible for conducting introductory programming labs in python for first-year engineering students. At that time, most of the students were first-time programmers, and my main challenge was to provide everyone with a fair chance to discuss the problems of programming labs and assignments. Through this experience, I have learned a few valuable lessons. Not everyone has the same resources and learns the same way. Therefore, I cannot teach everyone the same way and expect them to excel at the material. Significantly, the first time programmers may have difficulty expressing the issue, and I learned to clarify their doubts based on their needs patiently. I find students were particularly interested when solving a fun problem such as writing programs to draw shapes.

At Purdue University, I had the opportunity to be a teaching assistant for the **Introduction to Data Science (ECE 295)** class, and this was my very first time assisting an online course offering for engineers. I feel that when I explain something in-person from the instant feedback I get, I was able to recalibrate myself and clear the student's misunderstanding quickly. But in the online setup, since I did not have that instant feedback, I had to think twice before explaining anything. I had to come up with detailed but simple examples to explain concepts.

Instructor

In the summer of 2021, I had the privilege of teaching the **Data Structures (ECE 308)** class, and it is my first time teaching a class of 73 students. I had the chance to prepare course materials, deliver lectures, and evaluate the students. Usually, this is an in-person class. But due to the ongoing pandemic, I had to teach the class online.

My vision for this class was to equip students with skills that they may need for an algorithmic coding interview. I delivered the lectures via zoom and made the recorded video available for later viewing. Lecture slides were made available before the lectures. I like to write down on the slides (e.g., explaining running examples). These slides are also made available for the students. Even though I had allocated time for office hours, I had a virtual open-door policy because there were students in different time zones, and a big chunk of the class was on summer internships.

Over the course, I found certain teaching practices more effective than others to increase classroom engagement and improve students' understanding. For instance, students showed a greater engagement when I had semi-improvised live coding sessions in class and was given coding assignments based on real-world practical examples. In my view, you truly understand a concept when you can explain it well in simpler terms. One of my exams was akin to a coding interview where students were given a well-defined problem and asked for a solution in code. For the latter part of the exam, they were asked to solve a more challenging version of the previous problem, but they were not asked for a complete solution in code. Instead, they were asked to explain how they would solve the problem. Students were asked to submit a short video explaining their solutions. When students verbally explained solutions, I observed that they were self-correcting their mistakes in the code. Most students found this experience exciting and valuable.

Teaching Philosophy

Impactful teaching starts from taking initiatives to uphold inclusivity. I always like to maintain a fair, accessible, and collaborative classroom. Presumably, in such a classroom, everyone feels welcome to express their thoughts without hesitation. Especially underrepresented students may find it difficult to express themselves in front of everyone. Hence, I consciously try to keep an interactive environment by allocating time for discussion and encouraging questions from everyone. An inclusive classroom helps every student achieve their learning objectives while making teaching effective and efficient.

In my view, effective teaching provides both understanding and knowledge of the subject matter. It takes significant effort to design and implement material to understand the subject matter effectively. On the other hand, it is comparatively difficult to assess whether the students have acquired correct understanding when it comes to feedback and evaluation. Therefore, I develop my teaching philosophy around delivering a solid understanding of the subject matter. I believe that the following elements in teaching help construct a strong understanding of the concepts.

Well-established big picture of the subject: I always like to spend enough time explaining the bird's eye view of the subject. It helps to build an essential purpose for learning the material, and in turn, it strengthens the understanding.

Engaging and logically sequenced lectures: In my learning experience, cohesively presented subject matter has a high retention rate. I want to show a mind map of modules at the beginning of a course for students to understand how each module is connected to others. I revisit this mind map throughout the course to remind the big picture.

Fully encapsulated examples and analogies: When teaching complex concepts or recalling previously taught concepts, it is common to lose track. In some cases, too many details occlude the core concept and prevent the student from acquiring the correct understanding. Therefore, I like to structure my teaching materials around simplified examples and analogies whenever possible. This example makes teaching and studying easy and convenient to remind previously taught concepts.

Balanced assessment and evaluation: I find assessments that require regurgitating subject matter off-putting and skew the purpose of learning. Inspired from teaching the data structures class, I like to design a significant portion of the assessment focused on applying the knowledge learned to well-structured practical problems. This makes assessment more enjoyable for the students, especially those who prefer a hands-on learning experience.

Courses

The ability to teach a course is greatly enhanced when your area of research is closely related. I am broadly interested in compilers, programming languages, and high-performance computing. My research is focused on optimizing recursive programs operating on pointer-based data structures. I have the capacity to teach the following standard computer science courses:

Undergraduate-Level: Discrete Mathematics, Data Structures and Algorithms, Compilers, Programming Languages, and Parallel Programming

Graduate-Level: Compilers, High-Performance Computing, and Performance Engineering

Given the opportunity, I would like to design my own advanced topics course for graduate students focused on acquiring the knowledge for researching compilers and related topics.

Advanced Topics Course: Optimizing Compilers

A typical graduate-level course would cover the inner workings of a compiler, but it is too complex to go deep into any analysis or transformation. Also, modern compiler development has different styles with more and more functional programming concepts spilling over to the world of imperative programming. I would like to include the following topics focusing on optimizing compilers.

- Anatomy of a compiler pass.
- Static Single Assignment (SSA) form and Intermediate Representations (IRs).
- Advanced data-flow analysis (Partial Redundancy Elimination (PRE)).
- Modeling loops, recursion and transformation frameworks.
- Automatic parallelization.
- Learning techniques in compiler optimization and autotuning.
- Domain Specific Languages (DSLs) and optimizations.
- Multi-stage programming and program generators.
- Partial Evaluation (PE).

Conclusion

Teaching a subject that I deem I possess a thorough understanding of gives a brand new perspective. Productive strategies to communicate our understanding make us better teachers, and I develop such strategies using my past teaching experiences. My teaching philosophy is centered around strategies delivering the best understanding of the subject and making learning easy and enjoyable. In my humble opinion, teaching becomes impactful when we keep up inclusivity and accessibility.