From Theory to Practice and Back: A Tale of Combinatorial Designs, Optimization, and Search in Software Testing

Software is embedded in our daily lives, running business and home infrastructures, driving cars, and continuously, sometimes autonomously, making important decisions that impact a broad set of stakeholders. Yet, software is imperfect and prone to failures, potentially causing billions of dollars in financial losses each year. As such, research in software testing, has focused on developing novel ways to more efficiently and effectively find faults in programs before they escape into the field. The complexity and richness of modern programs, has led to the need for adaptation of techniques borrowed from other disciplines. For instance, there is a large community of researchers using SAT solving, evolutionary search-based optimization techniques, and learning to generate, select and prioritize test cases. While we can simply use these methods as tools, there is also an opportunity to extend and improve them based on what we learn from applying (or failing to apply) them to real systems. Software is an exciting domain for study, because it lacks the structure and symmetry of many synthetic problems.

In this talk I discuss several examples of how we have used combinatorial design theory, optimization and search for software test generation, and show how what we have learned in that process can potentially lead to novel research directions in those fields from which we have borrowed. First, I show how we can use combinatorial designs to improve software testing in practice, while using the same algorithms to improve bounds of theoretical interest to mathematicians. I then illustrate how the need for handling dependencies between software configuration options in real systems led to new constrained combinatorial objects and algorithms, and demonstrate how this led to a tight interleaving of heuristic search and SAT. I then present a hyper-heuristic search for software test generation that automatically learns and adapts over time, while producing potentially rich information about the underlying problem search space. Finally, I present some emerging software testing domains that we are studying, including some that are safety critical, where we are applying designs, optimization and search, to continue this interplay between practice and theory.