Covering arrays: new generalizations for software testing applications

Organizers:
Lucia Moura (University of Ottawa) and Brett Stevens (Carleton University)

Covering arrays are combinatorial designs that are used for testing systems such as software, circuits and networks, where failures can be caused by the interaction between their components or parameters. New generalizations of these objects employ techniques from design theory, graph homomorphisms, combinatorial group testing, among other fields. This minisymposium highlights current research that addresses some of the challenges that arise in real testing situations. Models under study incorporate graphs and hypergraphs to select relevant interactions to be tested, specification of forbidden interactions, and the ability to locate faulty interactions. This session will be closed by a talk discussing empirical data from applications and other challenges to be faced.

- **Adaptive algorithms for locating faulty interactions**
  Lucia Moura (University of Ottawa)
  Abstract:
  Locating arrays are recent generalizations of covering arrays that can determine the exact locations of faulty interactions in a system. In this talk, we look at pairwise interactions and represent faulty interactions by edges of a graph. Under certain assumptions on the structure of this graph, we give an efficient adaptive algorithm that locates all errors. The algorithm is able to handle some cases in which locating arrays do not exist. This is joint work with Conrado Martinez, Daniel Panario and Brett Stevens.

- **Covering arrays avoiding forbidden configurations**
  Peter Danziger (Ryerson University)
  Abstract:
  In this talk we consider designing covering arrays when certain pairs of factors are prohibited from our test suites. Such situations arise naturally when certain configurations of the test parameters are to be avoided. For example, in drug testing certain pairs of drugs may have known interactions and produce unwanted side effects. We show that in the case where the alphabet size $g > 2$, the problem of designing a covering array avoiding a specified configuration is NP complete. We also consider possible solutions for $g = 2$. This is joint work with Eric Mendelsohn, Lucia Moura and Brett Stevens.

- **Covering arrays on graphs**
  Karen Meagher (University of Waterloo)
  Abstract:
  In this talk I will describe a generalization of covering arrays, covering arrays on graphs. The original motivation for this generalization was for improving applications of covering arrays to testing systems and networks,

but this extension also gives us new ways to study covering arrays. In particular, the addition of a graph structure to covering arrays makes it possible to use methods from graph theory to study these designs. In this talk, I will describe a family of graphs called the *qualitative independence graphs*. Understanding these graphs will help understand covering arrays on graphs and standard covering arrays.

- **Constructions for optimal mixed covering arrays on graphs and hypergraphs**
  Christine Cheng (University of Wisconsin-Milwaukee)
  Abstract:
  Let $\mathcal{I}$ be a set of n parameters and $\mathcal{O}$ be a collection of subsets of $\mathcal{I}$. Suppose each $I \in \mathcal{I}$ has $\kappa(I)$ data values. A test case is an n-tuple $(t_1, t_2, \ldots, t_n)$, where t_i is a data value of I_i, $i = 1, \ldots, n$. In this talk, we present families of $(\mathcal{I}, \mathcal{O}, \kappa)$ for which an optimal test suite that covers all combinations of each $O \in \mathcal{O}$ (i.e., an optimal mixed covering array on a hypergraph) can be constructed efficiently.

- **Empirical results and practical extensions: using covering arrays to test configurable software**
  Myra Cohen (University of Nebraska-Lincoln)
  Abstract:
  Covering arrays can be used for software interaction testing to detect faults caused by combinations of configuration options or features. In this talk we present empirical studies that examine the effectiveness of covering arrays for testing highly configurable software systems. We discuss common features of these systems that render covering array models infeasible and suggest extensions to the model of a covering array to make them more applicable to the software testing application domain.