**General Algorithm Description:**

The algorithm has three main stages. First an initial range search is performed. The goal here is to find ranges of input parameters that seem promising and from which good test cases will likely be found. Then the main genetic algorithm commences, which attempts to evolve a population of test suites with diverse branch and multiple condition coverage. Finally a minimal sized, maximal coverage test suite is constructed from all the test cases across all organisms in the population. This final test suite is returned to the user, along with the percent of branch and multiple condition coverage achieved.

TODO: We should probably include not just total coverage ratio but a ratio for each branch and multiple condition in our final write up of experimental results.

**Global Range Set**

A set of promising input parameter ranges is maintained, from which the genetic algorithm can request new test cases during the initial population generation and test suite mutation steps. A range includes a start value and an end value, as well as an array of input parameter buckets. The buckets represent all the non-overlapping sub ranges of size 25 within the range. These buckets are mapped to counts of the number of times input parameters in those buckets were part of a test case that provided unique coverage population wide. The sum of all of these bucket counts forms a value referred to as the usefulness of that range.

**Range Set Initialization**

Before starting the genetic algorithm a simple procedure is run to find promising ranges of input values and add them to the global range set.

This procedure loops, starting with a range around 0 and extending in both directions towards max and min integer, generating a single test case from each range. These test cases are run through the control flow graph to determine which edges and predicates they cover. If a test case provides coverage that hasn’t yet been seen during this initial procedure, its corresponding range is added to the global range set. This procedure can be repeated using various range sizes, which in some cases enabled to algorithm to find additional coverage and ranges not found in previous iterations. However this is an expensive procedure whose goal is to simply identify promising ranges that help focus the genetic algorithm’s exploration. Thus for the purposes of the current research we found that performing two iterations, first with a range size of 5000, then with a range size of 2500 provided a good set of initial seed ranges for the genetic algorithm to utilize, while not taking too long to complete.

**Test Case Generation**

The range set provides two ways to generate new test cases. The first creates a test case by randomly selecting the values for all input parameters from a single range. This is used during the initial population generation described below.

The second uses repeated roulette wheel selection proportional to usefulness to select one range for each input parameter in the test case. In this way the algorithm generates test cases with input parameters stemming from the ranges that have been the most useful so far. The hope is that by combining parameters from several useful ranges into one test case, the new test case may provide coverage that would not be possible using parameters from just a single range. For example consider a multiple condition predicate that requires parameter 1 to be less than 500 but parameter 2 to be greater than 100000. In this situation ranges must be combined to form a test case that achieves the desired coverage.

**Range Set Adaptation**

After several generations of no improvement in coverage at the population level, the range set is adapted based on the usefulness of its constituent ranges. Ranges that have a usefulness value greater than one standard deviation above the mean are split into two equal sized ranges. In addition the two ranges above and below the original range are added to the range set. These new adjacent ranges have the same size as the original range. Ranges with a usefulness value less one standard deviation from the mean are deleted from the range set. Finally a new random range is added to the range set to avoid missing out on ranges not found during the range set initialization.

The idea behind this runtime adaptation is to give the algorithm the ability to learn which ranges are more useful than others, so that it can ultimately perform a more effective search for new test cases.

**Initialization**

As stated earlier the population of the proposed algorithm consists of test suites. Each test suite is initialized in such a way that all input parameters of all test cases in a given test suite are randomly selected from a single range. In this way ranges found during the initial range search form the initial niches to be explored by the algorithm.

**Test Case Operations**

The test case crossover and mutation operations provide the intra-niche search of the input parameters found across all test cases in a given test suite.

…