Chris Peabody

cgpzbd@mst.edu

COMP SCI 5401 FS2015 Assignment 1d

Description:

The multi-objective evolutionary algorithm in these experiments works on two objectives, MAXSAT fitness maximizing and robustness (minimizing the number of variables set). It does this by maintaining a data structure which symbolizes the layers of non-dominance among members of the population. In addition, parent and survival selection are based on the layers in which the population are stored. Fitness proportional selection, for example, is now proportional to the layer of non-dominance the individual belongs to. The closer to the first layer the individual is, the more of a chance it has to be selected. Similarly, k-tournament selection and truncation survival selection are based off of the non-dominance layers as well, rather than the fitness of the individuals.

In order to facilitate unset variables, support for a value other than True or False was added. This value served as a place-holder representing an unset variable. When uniform crossover occurred, if the parent which was selected to pass on an allele had the unset value, the child created received that allele, making it unset for it as well.

In addition to initialization and recombination supporting non-set variables, the mutation operator also allowed variables to mutate to the unset state. It also allowed for unset variables to mutate into True and False values.

After these major distinctions were made, and output files and analysis tools were changed to represent them, the MOEA operated just like a traditional EA.

Experiment:

In the experiment, 5 tests were executed, each with a different configuration file. The first configuration file was set as the baseline, and each file after that was a modified version of it. Each test consisted of 30 runs, at 10,000 evaluations each.

Test 1: The first configuration file used fitness proportional parent selection, truncation survival selection, and a standard plus strategy with Mu = 100 and Lambda = 50.

Test 2: The second configuration file matched the first, except r-elitist restarts were enabled, with r = 50.

Test 3: The third configuration was the same as the first, except a comma strategy was used, and Lambda was changed to 200 to facilitate this.

Test 4: The fourth configuration mirrored the first, except k-tournament selection was used in place of proportional selection. K was set to 10 for this.

Test 5: The fifth configuration was the same as the first, except Mu and Lambda were doubled. Mu = 200, Lambda = 100.

Results:

Each of the configuration files were tested and the resulting pareto fronts were compared to one another. In addition, the diversity of each resulting pareto front was tested. The diversity of each test can be seen in Table 1 below. The values are based on the volumes of hyper-cubes between neighboring solutions in the curve. A diversity measure closer to 1 indicates the set of individuals is more evenly spread along the curve, while a lower score means a poor distribution. The diversity measure of the first two tests were very similar. They were the most diverse of the solutions found. Test 3 was the least diverse of the solutions.

	Diversity Measure
Test 1	0.932712676
Test 2	0.936920188
Test 3	0.865488263
Test 4	0.896264789
Test 5	0.887784977

Table 1: Diversity of the resulting pareto fronts for each test. Values closer to 1 mean more diverse solutions, while values closer to 0 mean less diverse solutions.

The solutions of each test were analyzed against each other. A set of data for each test was created to represent what percentage of the time each a run in the algorithm dominated a run in the opposing algorithm. This information was then used to rate the algorithms against each other. Several of the data sets created had all of their runs dominating 100% of the opposing algorithm, while each run of the opposing algorithm dominated 0% of the first data set. It is clear that in these cases, the algorithm with all the 100% data points is better than the algorithm with the 0% data points for these experiments.

The author used this information to first create an initial hierarchy of which tests ran better than which. The hierarchy can be seen in Table 2.

Level	Test(s)
1	Test 1, Test 2
2	Test 4
3	Test 3, Test 5

Table 2: Every test in each level had 100% domination on each run for tests in lower levels (I.e. Test 1 had 100% domination over tests 3, 4, and 5). Statistical analysis was still needed for tests within the same level of the hierarchy, since their percentages were not cut and dry.

The author first compared test 1 and 2. An f-test for equal variances concluded that you could not assume equal variances. The author then conducted a t-test assuming unequal variances. The result suggested accepting the null hypothesis, where it cannot be assumed the test with a higher mean is statistically better. Because of this, it was concluded Test 1 and Test 2 were about equal in quality. Adding in r-elitist restarts had a minimal effect on the algorithm.

The author then compared test 3 to test 5. The f-test for equal variances concluded that you can assumed equal variances. The t-test which followed suggested rejecting the null hypothesis, which allows you to assume the algorithm with the better mean percentages was, in fact, better. Test 5 had a

mean percentage of 86, while test 3 had a mean percentage of 11. The author concluded test 5 was better than test 3.

Conclusion:

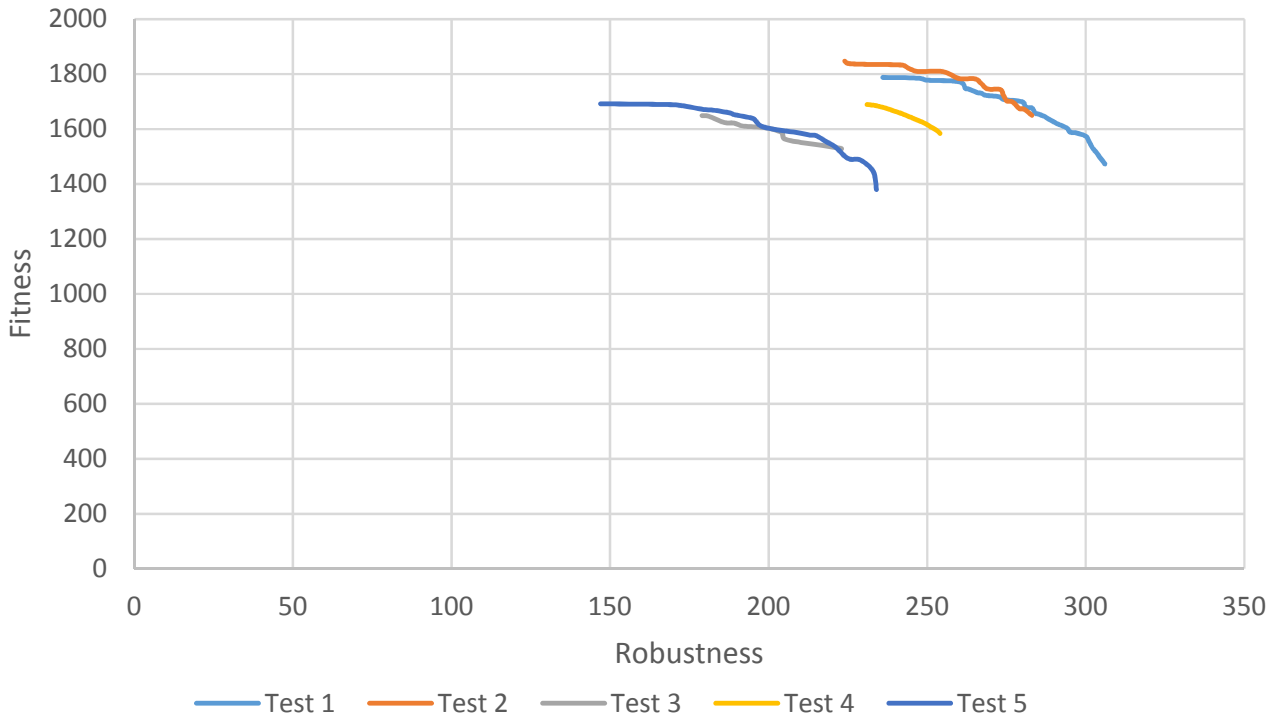
Test 1 and 2 were on par with one another. Adding in r-elitist restarts did not drastically effect the productivity of the algorithm. These two tests showed better results, as well as a higher diversity measure than all the other configurations.

The next best configuration was in test 4. This test used k-tournament selection rather than proportional selection. It's resulting pareto front also had the next highest diversity measure.

Test 5 was the next best solution and next best diversity measure. Doubling Mu and Lambda had a negative effect on the productivity of the algorithm. Interestingly enough, Test 5 also took much longer to run that the other tests. The author believe this to be because of the exponential nature of the sorting algorithm used to construct the layers of non-dominance.

Test 3 was the worst of the solutions and the worst of diversity. Using a generational strategy had an impacting negative effect on the algorithm. This test was the fastest to run. The author believes this to be true because on each generation the program only had to select from 50 more individuals than other configurations, while finishing 4 times as many evaluations at each generation.

Each test's pareto fronts were graphed against one another, and can be seen in Graph 1 on the next page. Tests which completely dominated one another reflect this on the graph.



Graph 1: Pictured in this graph are the best pareto fronts found in any given test. The pareto fronts are curves created on their level of fitness and robustness and are non-dominated in their given