Adaptive Random Testing with CG Constraint

F. T. Chan
School of Professional and Continuing Education,
The University of Hong Kong
ftchan@hkuspace.hku.hk

T. Y. Chen
School of Information Technology
Swinburne University of Technology
tychen@it.swin.edu.au

K. P. Chan *
Department of Computer Science & Information Systems
The University of Hong Kong, Pokfulam Road, Hong Kong.
kpchan@csis.hku.hk

S. M. Yiu
Department of Computer Science & Information Systems
The University of Hong Kong
smyiu@csis.hku.hk

Abstract

In this paper, we introduce a C. G. constraint on Adaptive Random Testing (ART) for programs with numerical input. One rationale behind Adaptive Random Testing is to have the test candidates to be as widespread over the input domain as possible. However, the computation may be quite expensive in some cases. The C. G. constraint is introduced to maintain the widespreadness while reducing the computation requirement in terms of number of distance measures. Three variations of C. G. constraints and their performance when compared with ART are discussed.

Keywords

Random Testing, Adaptive Random Testing, Center of Gravity constraint

1. Introduction

Selection of test cases is the most basic problem in software testing. There are two approaches towards the selection of test cases: white box and black box approaches. Among the black box approaches, Random Testing is one of the most common techniques employed. The main merits of random testing include the availability of efficient algorithms to generate test cases and its ability to infer reliability and statistical estimates [1, 2].

Recently, the technique of adaptive random testing [3] has been developed as a failure pattern oriented random testing method, where inputs are randomly selected as test cases as long as they are not close to the already executed ones. Though the modification is quite simple and easily implemented, the improvement of fault detection capability is quite amazing. In some empirical studies, We require 40% less test candidates to detect the first failure when compared with Random Testing. However, adaptive random testing sometimes requires extensive computation in order to ensure that the selected test cases are not close to the already executed ones.

2. Adaptive Random Testing

The rationale behind ART is that test cases, while randomly generated, should be as evenly spread over the entire input domain as possible. In a simple implementation of ART, a number of potential candidates are first randomly generated. Then the el-

*All correspondence should be addressed to Dr. K. P. Chan
ment that is “farthest away” from the executed set is chosen and tested.

There are many possible ways to define farthest away. For example, Chen et. al. [3] uses
\[
\min_{i=1}^{n} \text{dist}(c_i, t_i) \geq \min_{j=1}^{n} \text{dist}(c_j, t_i)
\]
(1)
where \(c_i\) is the chosen candidate, \(c_j\) is a candidate generated, and \(t_i\) is an already tested candidate. Chan et. al [6, 7, 8] use the notion of exclusion, and come up with the Restricted Random Testing. There are many other ways that farthest away, wide spreadness and evenly distributed can be defined. The C.G. constraint is another direction in this approach.

Chen et. al. applied the ART algorithm on 12 error-seeded programs. These programs are published programs, all involving numerical computation [9]. Various kinds of errors are seeded into the programs, including arithmetic operator replacement, relational operator replacement, scalar variable replacement and constant replacement. It was found that the ART may achieve an improvement up to about 50% over Random.

One drawback of ART is its extensive computation requirement. During each iteration, the distance between each element in the candidate set and every element in the executed set is needed. When the testing continues, the number of distance calculation increases significantly. We would like to introduce other criteria so that we can maintain the executed set to be as wide-spread in the input domain as possible while reducing the computation requirement. This leads to the use C.G. constraint. In the following sections we will discuss possible formulations of C.G. constraint.

3. Combining various Testing Strategies

It is well known in Pattern Recognition that combining various recognizers together can improve recognition rate. For example, Boosting [10] is a very popular such approach. Combining testing strategies together is also appealing theoretically, as the F-measures of different strategy usually has a very large variance, because of the random nature of test candidate selection. In fact, in a theoretical model of Restricted Random Testing (RRT) [8], the magnitudes of the standard deviation and F-measures are of similar values. This means that the range of the F-measures can be very large, and the actual F-measure for a particular iteration can even be several times that of the mean F-measure. Hence combining different strategies is appealing.

However, if we apply strategy A and B alternately, the F-measure will double the minimum of the two. Instead of applying them alternately, we use a probability distribution. The probability of applying one strategy can be based on the performance of the strategy when applied alone. Furthermore, the computation requirement of the resultant combined strategy will be reduced, as the strategy with higher computation requirement (which usually produces better results) will be applied only according to the probability distribution.

4. C.G. constraint

The center of gravity of a set of points is defined as the average of the points. We will expect the points in the set to be evenly distributed around the C.G. Also, the C.G. can be updated efficiently, without finding the average of all tested candidates set each time, by
\[
\tilde{\text{g}}_{i+1} = \frac{\bar{x} + \text{g}_i}{n + 1}
\]
(2)
where \(\text{g}_i\) is the C.G. at the \(i\)-th iteration, \(x\) is the next chosen test case, and \(n\) is the number of tested candidates.

This points us to the naive C.G. constraint: the test case is chosen from the candidate set so that the resultant C.G. is to be as close to the C.G. of the testing domain as possible. However, such straightforward application of the C.G. constraint may result to a so-called Black-hole effect. Since the resultant C.G. should be close to the center of the input domain at each iteration, the chosen test candidate will also be close to the center. This will cause a clustering of chosen test candidates around the center of the input domain, which is certainly undesirable.

To escape from the black hole effect, we can combine the ART together with C.G. constraint. This allows us to introduce candidates randomly generated from the ART together with candidates from the C.G. constraint, and hence avoid the black-hole. Instead of a fixed ratio between ART and C.G. we generate candidates probabilistically, according the distribution:
\[
\begin{align*}
\text{Prob. (candidates by ART)} &= p \\
\text{Prob. (candidates by C.G.)} &= 1 - p
\end{align*}
\]

4.1. Variation of the C.G. constraint

4.1.1. Closest C.G. constraint

In this formulation, we require the resultant C.G. to be as close as a target C.G. which is usually chosen to be the C.G. of the input domain. This can ensure the members in the executed set will be as evenly distributed around the target C.G. as possible. However, when applied alone, the constraint will lead to Black-hole effect as mentioned above, and it has to be applied together with other random methods.

4.1.2. Farthest C.G. Constraint

Instead of requiring the resultant C.G. to be as close to a target C.G. as possible, we require the C.G. to be as far away as possible from the previous C.G. instead of a predefined target C.G. In this way, the candidates chosen will be far away from the existing candidates in the executed set.

4.1.3. Direct C.G. Constraint

Instead of forming a candidate set and choose one candidate from this set based on C.G. constraint, we generate a candidate so that the resultant C.G. will be at the center of the testing domain. In order to preserve the essence of randomness, the test candidate
is chosen such that it is randomly picked within a window centered at the candidate which will make the resultant C.G. to lie exactly on the center of the input domain. Again, we need a combination of ART and direct C.G. constraint in order to avoid the black-hole effect.

### 4.2. Computation Requirement of C.G. constraint

At each testing iteration, the new C.G. can be computed by equation 2. For Closest and Farthest C.G. we need to compute each distance between the resultant C.G. of each of the candidates in the candidate set, and the target C.G. or the previous C.G. Hence it involves only \( k \) distance calculation in each iteration, where \( k = 1 \) is a constant, which is chosen to be 10. For Direct C.G. no distance measure is needed as we generate the candidate directly.

Consider at iteration \( L \), we will have about \( pL \) elements in the executed set generated by ART, while the rest from C.G. The expected number of distance measure required in this iteration is then given by

\[
d_L = pkL + (1 - p)k
\]

Hence total number of distance measure required for a testing going on for \( N \) iterations:

\[
D = \sum_{L=1}^{N} d_L = \sum_{L=1}^{N} pkL + (1 - p)kN
\]

To limit the amount of computation, we can also include only those test cases in the executed set that are generated by ART to be used. The corresponding computation requirement is

\[
D' = p^2kN(N - 1)/2 + (1 - p)kN
\]

We called the latter C.G. fast version.

### 5. Empirical Study

We combine the ART method with the 3 C.G. constraint, using \( p = 0.667 \) and \( p = 0.5 \), i.e. about two-thirds, or half of the candidates will be generated from ART. We applied both the C.G. and the C.G. fast version in the testing. For the direct C.G. constraint, we use a window size of 5% of the input domain along each dimension, in which the test case will be chosen.

The testing strategies were applied to one of the program, cel, used in the ART study as a preliminary investigation. The input of the cel program is 4-dimension, and 3 errors, namely, arithmetic operator replacement, relational operator replacement and constant replacement are seeded. The failure rate for this program is 0.000039. The \( F \)-measures of different testing strategies was obtained empirically. Each testing strategy was applied 1000 times, and the average \( F \)-measure is computed. Figure 1 shows the average \( F \)-measures using the ART, and combined ART with 3 C.G. constraint at \( p = 0.667 \) and \( p = 0.5 \). The lower line is the \( F \)-measure of ART and the upper line that of Random Testing, which serves as the upper and lower bound for \( F \)-measures. Table 1-4 summarizes the statistics of the experiment.

<table>
<thead>
<tr>
<th>( \text{Table 1} ). Statistics of ART and C.G. with ( k = 10 ) and ( p = 0.667 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ART} ) &amp; CG-Far &amp; CG-Close &amp; CG-Direct</td>
</tr>
<tr>
<td>Average ( F ) &amp; 1598.1 &amp; 1778.2 &amp; 1825.5 &amp; 2335.2</td>
</tr>
<tr>
<td>S.D. &amp; 1577.7 &amp; 1729.7 &amp; 1685.2 &amp; 2236.6</td>
</tr>
<tr>
<td>Max ( F ) &amp; 11356 &amp; 10409 &amp; 11166 &amp; 15121</td>
</tr>
<tr>
<td>Min ( F ) &amp; 4 &amp; 1 &amp; 2 &amp; 2</td>
</tr>
<tr>
<td>Times better &amp; - &amp; 479 &amp; 482 &amp; 417</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \text{Table 2} ). Statistics of ART and C.G. with ( k = 10 ) and ( p = 0.667 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ART} ) &amp; CG-Far &amp; CG-Close &amp; CG-Direct</td>
</tr>
<tr>
<td>Average ( F ) &amp; 1598.1 &amp; 1977.5 &amp; 1904.5 &amp; 2878.7</td>
</tr>
<tr>
<td>S.D. &amp; 1577.7 &amp; 1942.9 &amp; 1810.7 &amp; 2669.7</td>
</tr>
<tr>
<td>Max ( F ) &amp; 11356 &amp; 15961 &amp; 15507 &amp; 17158</td>
</tr>
<tr>
<td>Min ( F ) &amp; 4 &amp; 6 &amp; 1 &amp; 1</td>
</tr>
<tr>
<td>Times better &amp; - &amp; 458 &amp; 444 &amp; 366</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \text{Table 3} ). Statistics of ART and C.G. fast with ( k = 10 ) and ( p = 0.667 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ART} ) &amp; CG-Far &amp; CG-Close &amp; CG-Direct</td>
</tr>
<tr>
<td>Average ( F ) &amp; 1598.1 &amp; 2048.6 &amp; 2143.3 &amp; 2443.1</td>
</tr>
<tr>
<td>S.D. &amp; 1577.7 &amp; 2087.9 &amp; 2104.4 &amp; 2434.6</td>
</tr>
<tr>
<td>Max ( F ) &amp; 11356 &amp; 13986 &amp; 17088 &amp; 23353</td>
</tr>
<tr>
<td>Min ( F ) &amp; 4 &amp; 3 &amp; 4 &amp; 5</td>
</tr>
<tr>
<td>Times better &amp; - &amp; 423 &amp; 434 &amp; 393</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \text{Table 4} ). Statistics of ART and C.G. fast with ( k = 10 ) and ( p = 0.5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ART} ) &amp; CG-Far &amp; CG-Close &amp; CG-Direct</td>
</tr>
<tr>
<td>Average ( F ) &amp; 1598.1 &amp; 2086.7 &amp; 2100.2 &amp; 2474.4</td>
</tr>
<tr>
<td>S.D. &amp; 1577.7 &amp; 2087.9 &amp; 2075.2 &amp; 2481.2</td>
</tr>
<tr>
<td>Max ( F ) &amp; 11356 &amp; 13012 &amp; 14461 &amp; 21907</td>
</tr>
<tr>
<td>Min ( F ) &amp; 4 &amp; 8 &amp; 5 &amp; 11</td>
</tr>
<tr>
<td>Times better &amp; - &amp; 411 &amp; 420 &amp; 378</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \text{Table 5} ). Comparison of testing strategy with and without reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterations &amp; ART &amp; CG-Far &amp; CG-Close &amp; CG-Direct</td>
</tr>
<tr>
<td>500 &amp; 0.277 &amp; 0.277 &amp; 0.24 &amp; 0.24</td>
</tr>
<tr>
<td>1000 &amp; 0.472 &amp; 0.477 &amp; 0.44 &amp; 0.422</td>
</tr>
<tr>
<td>1500 &amp; 0.613 &amp; 0.622 &amp; 0.565 &amp; 0.561</td>
</tr>
<tr>
<td>2000 &amp; 0.713 &amp; 0.727 &amp; 0.666 &amp; 0.666</td>
</tr>
<tr>
<td>2500 &amp; 0.787 &amp; 0.802 &amp; 0.756 &amp; 0.746</td>
</tr>
<tr>
<td>3000 &amp; 0.852 &amp; 0.857 &amp; 0.814 &amp; 0.807</td>
</tr>
</tbody>
</table>
6. Discussion

From the results shown in figure 1, we found that ART was still the best performer in terms of F-measures. Among the various C.G. constraints, the result of Closest C.G. and Farthest C.G. are very close to each other. Also, we can notice that the \( p = 0.667 \) performs better than \( p = 0.5 \) for the ordinary version, while their performance difference is not very large. Hence there will be a tradeoff between computation requirement and performance in terms of F-measure. Furthermore, we found that by including C.G. constraint, the F-measure only increase by about 11% (for C.G.-Far with \( p = 0.667 \)), while the computation can be reduced. For example, when \( p = 0.667 \), the F-measure of the C.G. strategy is 1.11F, where F is the F-measure of ART, and the computation requirement of the combined strategy is \( 0.667 \times 1.11 = 0.74 \) of the original ART. Furthermore, the F-measure is still much lower than that of the Random Testing strategy. For \( p = 0.5 \), the computation requirement is just 0.57 of the original. For the fast version, the F-measure is about 30% higher than ART. Hence the computation will be \( 0.667 \times 1.3 = 0.58 \) of the original ART. Similarly, for \( p = 0.5 \), the ratio is reduced to around 0.33.

Figure 1. Graph of F-measures of ART and C.G. constraint

From Table 1, we find that out of 1000 times, the C.G. constraint actually finds the first failure faster than ART in about 480 times. We also notice that the distribution is a skew distribution, with about 28% F-measures lying in the first 0 and 500 for ART, and about 24% for C.G.-Far, and the distribution drops when F-measure increases. A recent study has found that F-measure follows a geometric distribution [11].

One possible way to make use of this observation is to reset the C.G. constraint regularly, so that if the testing drag for too long, it will start afresh. Table 5 shows the detail distribution of the F-measures in the 1000 iterations. We can find that the probability of finding the first failure in the first 500 and the next 500 iterations are 0.277 and 0.195 respectively, and the probability decreases rapidly afterwards. One possible change to the algorithm is that we can reset the testing after some fixed number of trials, \( n \). Suppose the probability of finding the failure in the first \( n \) trial is \( p \). After \( n \) trials, we restart the testing. Then the probability of not finding the failure after \( kn \) trials is given by \( (1 - p)^k \), and hence the probability of finding the failure within the first \( kn \) trial is \( 1 - (1 - p)^k \). Table 5 shows the comparison between the original distribution and the calculated reset probability.

From this table, we can observe that the two probabilities are very close to each other. Since the number of distance measure required is \( O(n^2) \), we will have much lower computational requirement.

7. Conclusion

In this paper, we propose an extension of the ART by introducing the C.G. constraint. We combine the ART with C.G. constraint by a probability distribution. Empirical result shows that the F-measure of C.G. constraint is about 11% higher than ART, while the computation required is about 74% of it.

Acknowledgement

We should like to gratefully acknowledge the support given to this project by the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No: HKU 7007/99E).

References