Controlling Restricted Random Testing: An Examination of the Exclusion Ratio Parameter

Kwok Ping Chan
Dept. of Comp. Sci.,
The Univ. of Hong Kong
Pokfulam Road,
Hong Kong, China
kpchan@cs.hku.hk

T. Y. Chen
Faculty of Information & Communication Technologies,
Swinburne Univ. of Technology,
Hawthorn 3122, Australia
tychen@it.swin.edu.au

Dave Towey *
Division of Sci. & Tech.,
BNU–HKBU United International College, Tangjiawan,
Zhuhai, 519085, China
davetowey@uic.edu.hk

Abstract

In Restricted Random Testing (RRT), the main control parameter is the Target Exclusion Ratio (R), the proportion of the input domain to be excluded from test case generation at each iteration. Empirical investigations have consistently indicated that best failure-finding performance is achieved when the value for the Target Exclusion Ratio is maximised, i.e. close to 100%. This paper explains an algorithm to calculate the Actual Exclusion Ratio for RRT, *and applies the algorithm to several simulations, confirming that previous empirically determined values for the Maximum Target Exclusion Ratio do give Actual Exclusion Ratios close to 100%. Previously observed trends of improvement in failure-finding efficiency of* RRT *corresponding to increases in Target Exclusion Ratios are also identified for Actual Exclusion Ratios.*

KEYWORDS: Software Testing; Random Testing; Adaptive Random Testing; Restricted Random Testing; Exclusion Ratio.

1. Introduction

Random Testing incorporating additional mechanisms to ensure more widespread distributions of test cases over an input domain have been called Adaptive Random Testing (*ART*) [2, 3, 4, 5, 6, 7]. It is considered a promising direction of automatic test case generation [8].

A version of *ART*, based on the use of exclusion, is the Restricted Random Testing (*RRT*) method [2]. By excluding regions surrounding previously executed test cases, and restricting subsequent cases to be drawn from other areas of the input domain, *RRT* ensures an even distribution, and guarantees a minimum distance amongst all cases. In experiments, the *RRT* method has outperformed *RT* by up to 80% on some occassions.

It has been observed that the failure-finding efficiency of *RRT* improved as the Target Exclustion Ratio (*R*) was increased, with the best failure-finding efficiency achieved when *R* was at a maximum [2]. The *Max R* refers to the maximum value for *R* beyond which the Actual Exclusion Ratio is too close to 100% for test cases to be generated.

The difference between Target and Actual exclusion is due to (1) *Overlapping* (*Olp*) of exclusion regions; and (2) portions of the exclusion regions falling *Out the Input Domain* (*OID*). Because of the importance of the *Max R*, the ability to accurately determine the Actual Exclusion Ratio for a given Target Exclusion Ratio was desirable.

In this paper, we explain an algorithm to calculate the Actual Exclusion Ratio and give the results of an application of the algorithm to estimate the expected Actual Exclusion Ratio for a given Target Exclusion Ratio. These results confirm that the

* Corresponding author

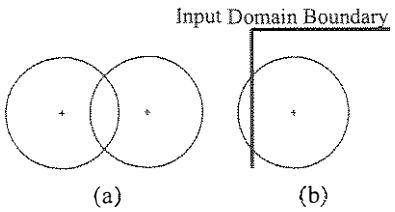


Figure 1. (a) Example of *Overlapping* (*Olp*) of exclusion regions (b) Example of portion of an exclusion region falling *Outside the Input Domain* (*OID*)

Actual Exclusion for Target Exclusion Ratio values near *Max R* is close to 100%. They also show that the Actual Exclusion does increase as Target Exclusion increases.

2. Maximum Target Exclusion

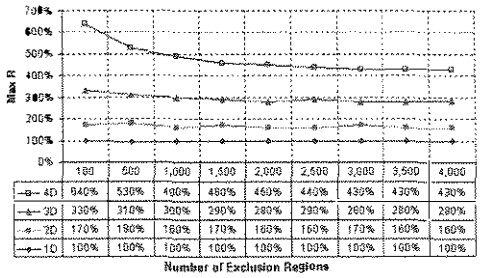
In *RRT*, given a test case that has not revealed failure, the area of the input domain from which subsequent test cases may be drawn is restricted. By employing a hyperspherical zone, a minimum distance (the radius of the exclusion zone) between all test cases is ensured.

All exclusion zones are of equal size, and this size decreases with successive test case executions. The size of each zone is related to both the size of the entire input domain, and the number of previously executed test cases.

The final (and most important) determinant of exclusion zone size in *RRT* is the Exclusion Ratio (*R*). This figure is applied to the total area of the input domain to obtain the total exclusion area.

During the execution of the *RRT* algorithm, the Actual Exclusion Ratio is usually less than the Target Ratio. This occurs when there is *Overlapping* (*Olp*) of exclusion regions (Fig. 1(a)); or when some portion of an exclusion region falls outside of the Input Domain, (*OID*) (Fig. 1(b)); or when some combination of both these situations occurs.

We defined the maximum target exclusion ratio (*Max R*) as the highest *R* at which it is still possible to generate test cases for a full sample size, *n*. Because of the difficulty of an analytical investigation of *Max R*, simulations were run to investigate what factors influenced it. In the simulations, the



	100	500	1,000	1,500	2,000	2,500	3,000	3,500	4,000
4D	540%	530%	490%	480%	460%	440%	430%	430%	430%
3D	330%	310%	300%	290%	280%	290%	280%	280%	280%
2D	170%	190%	180%	170%	180%	160%	170%	160%	160%
1D	100%	100%	100%	100%	100%	100%	100%	100%	100%

Figure 2. Results of Maximum Target Exclusion Rate (*Max R*) calculation for a homogeneous input domain. Sample size (n) is 100, and number of exclusion regions varies from 100 to 4,000

number of regions was varied from 100 to 4,000, the sample size (*n*) was set at 100, and *R* was incremented by 10% each time. A limit of 100,000 on the number of attempts to generate a valid test case was imposed for each test case. The input domain shape was homogeneous (square in 2D, cube in 3D, and hypercube in 4D). The results are summarized in Fig. 2. They indicate that, when the number of exclusion regions is lower, the maximum target exclusion (*Max R*) is higher.

Because of the importance of the relation between Actual and Target Exclusion, an algorithm to calculate the Actual Exclusion Ratio was developed. As the best failure-finding performance was obtained when circular regions were used [1], our algorithm calculates Actual Exclusion for circular exclusion regions.

3. Actual Exclusion Ratio Calculation

In this section, the algorithm for calculating the Actual Exclusion Ratio is explained. The method examines the distribution of test cases and their exclusion regions, and calculates the loss of exclusion region area caused by *Overlapping* (*Olp*) of regions, and by portions of regions falling *Outside the Input Domain* (*OID*).

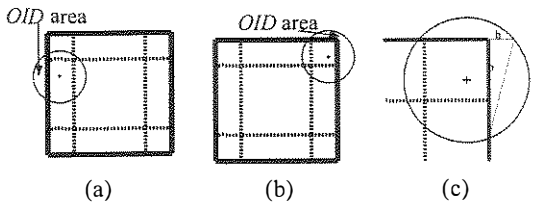


Figure 3. (a) *OID* when TC is in a non-comer region of the *gutter* (b) *OID* when TC is in a corner region (c) Partitioning of *OID* for area calculation

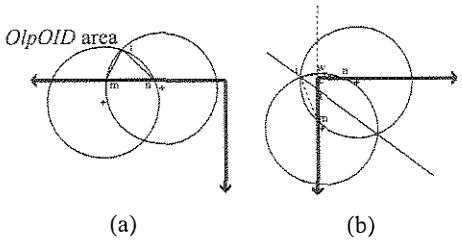


Figure 4. (a) Example of *OlpOID* and Input Domain on only one border (b) Example of *OlpOID* and Input Domain on two borders

3.1. Actual Exclusion Ratio Calculation Algorithm

The Actual Exclusion will differ from the Target Exclusion when any combination of the following occurs: (a) *OID* — occur when any previously executed test case (TC_x) lies within a distance r of the input domain border (this area of depth r inside the border is referred to as the *gutter*). (b) *Olp* — occur when any previously executed test cases (TC_x, TC_y) are within twice the exclusion zone radius of each other.

To calculate the Actual Exclusion Ratio, we first find the Target Exclusion Area, and then subtract the total area of *OID*, and then subtract the area of *Olp* inside the Input Domain.

3.2. Calculation of OID

First, the location of each previously executed test case is examined to determine whether it will have any portion of its exclusion zone lying outside the Input Domain. This occurs when the TC is within distance r (exclusion radius) of the Input Domain border. There are two cases.

The *OID* area for the TCs not lying in the corners of the *gutter* can be straightforwardly calculated from the area of the *circle segment* formed when the Input Domain border, acting as the *secant*, cuts the exclusion region circle.

Calculation of the *OID* area for the TCs which are in the corners of the *gutter* requires partitioning the *OID* region into three smaller regions, as shown in Fig. 3(b). Fig. 3(c) shows how this area is partitioned into a triangle and two *circle segments*.

3.3. Calculation of Olp

First, every pair of executed test cases is examined to see if there is any overlap between their exclusion regions. The overlap is calculated by bisecting the region with the line through the intersections of the circles, creating 2 identical *circle segments*.

Next, the location of the circles' intersection points (*ip*) is checked. If an intersection point is outside the Input Domain, some of the *Olp* area will also lie outside, and must be subtracted from the total *Olp* loss. The portion of *Olp* lying Outside the Input Domain is referred to as *OlpOID*.

For an intersection point (*ip*) outside the Input Domain, the area of the Overlap lying Outside the Input Domain (*OlpOID*) is calculated in one of two ways, according to whether the *OlpOID* area touches the Input Domain on one or two borders.

3.4. Calculation of OlpOID

When the *OlpOID* is on only one border (Fig. 4(a)), the area can be calculated by totaling the areas of the triangle formed by the intersection of the circles (*i* in Fig. 4(a)) and the intersections of the *arcs* from *i* to the border (*m* and *n*), and the two *circle segments* formed by the *chords* from the intersection of the circles to the intersections of the *arcs* and the border.

When the *OlpOID* is on two borders, the region is split into three triangles and three *circle segments*, as shown in Fig. 4(b). The area of these regions can be easily calculated.

Table 1. Target vs Actual exclusion ratios for *RRT*.

Target Excl. Ratio	Actual Exclusion (%) Total Number of Exclusion Zones							
	10	50	100	150	500	1000	5000	10000
1%	0.98	0.99	0.99	0.99	1.00	1.00	1.00	1.00
20%	18.21	18.87	19.05	19.12	19.28	19.34	19.41	19.43
40%	34.15	36.03	36.51	36.72	37.16	37.29	37.50	37.55
60%	48.05	51.33	52.15	52.55	53.26	53.52	53.89	53.98
80%	60.22	64.62	65.69	66.17	67.20	67.59	68.09	68.21
100%	70.16	75.48	76.76	77.33	78.52	78.95	79.50	79.64
120%	78.29	83.69	84.97	85.51	86.61	86.97	87.47	87.58
140%	84.60	89.12	90.07	90.42	90.98	91.15	91.33	91.36
160%	88.72	91.68	91.88	91.75	91.37	91.12	90.70	90.57

4. Actual Exclusion Ratio

The algorithms described were applied to several simulations, varying the Target Exclusion Ratio, and the total number of executed test cases. The simulations were conducted within a square input domain.

Table 1 shows the results for different numbers of exclusion regions (10 to 10,000), averaged over 1,000 trials. In the table, Target Exclusion is the percentage area of the Input Domain which we attempt to exclude from random point generation, this was varied from 1% to 160%. Actual Exclusion is the average percentage of the Input Domain which is actually excluded by the exclusion zones.

As expected, there is a difference between the Target and Actual Exclusion, and this difference appears to become more pronounced as the number of test cases increases, and also as the Target Exclusion Ratio increases. It also confirms that the Actual Exclusion ratios increase with R, i.e., the improvement in failure-finding efficiency does correspond to increases in the Actual Exclusion.

By nature, *Max R* may vary with the sample size. Table 1 shows that around the *Max R* values, the Actual Exclusion Ratio is very close to 100%. However, with high Actual Exclusion, the number of attempts necessary to generate a test case outside the exclusion regions increases: e.g. 99% exclusion leaves only 1% of the input domain outside the exclusion regions, which would take an average of 100 (1/1%) trials to find.

5. Summary

In this paper, we presented an algorithm for calculating the Actual Exclusion Ratio, in 2D, for the *RRT* method. We also presented the results of simulations for various numbers of test cases, and varying Target Exclusion Ratios. These results confirmed that the Actual Exclusion does increase as Target Exclusion increases. Hence, the improvements for failure-finding efficiency are linked to increases in Actual Exclusion. Also, the Actual Exclusion Ratio for Target Exclusion Ratio near *Max R* is indeed close to 100%. The results provide stronger theoretical support for *RRT*.

Acknowledgment

This project is partially supported by an Australian Research Council Discovery Grant (DP0557246).

References

[1] K. P. Chan, T. Y. Chen, and D. Towey, "Adaptive Random Testing with Filtering: An Overhead Reduction Technique", *17th Int. Conf. on Software Engg. and Knowledge Engg. (SEKE'05)*, Taipei, Taiwan, July 14-16, 2005.

[2] K. P. Chan, T. Y. Chen, and D. Towey, "Restricted Random Testing: Adaptive Random Testing by Exclusion", *Int. J. of Software Engg. and Knowledge Engg.*, Vol. 16, No. 4, pp. 553–584, 2006.

[3] T. Y. Chen, F. C. Kuo, R. G. Merkel, S. P. Ng, "Mirror Adaptive Random Testing", *Inf. and Software Tech.*, Vol. 46, No. 15, pp. 1001–1010, 2004.

[4] T. Y. Chen, H. Leung, and I. K. Mak, "Adaptive Random Testing", *Lecture Notes in Comp. Sci.*, Vol. 3321, pp. 320–329, Jan 2004.

[5] J. Mayer, "Adaptive Random Testing by Bisection and Localization", *5th Int. Workshop on Formal Approaches to Testing of Software (FATES 2005)*, LNCS 3997, Springer-Verlag, pp. 72–86, 2006.

[6] J. Mayer, "Adaptive Random Testing by Bisection with Restriction", *Formal Methods and Software Engg, 7th Int. Conf. on Formal Engg. Methods (ICFEM 2005)*, LNCS 3785, Springer-Verlag, pp. 251–263, 2005.

[7] J. Mayer, "Lattice-Based Adaptive Random Testing", *20th IEEE/ACM Int. Conf. on Automated Software Engg.*, pp. 333–336, 2005.

[8] I. Ciupa, A. Leitner, M. Oriol, and B. Meyer, "Object Distance And Its Application To Adaptive Random Testing Of Object-Oriented Programs", *1st International Workshop on Random Testing (ISSTA-RT 2006)*, pp. 55-63, 2006.