Forgetting Test Cases

Kwok Ping Chan
Department of Computer Science,
The University of Hong Kong,
Pokfulam Road,
Hong Kong SAR,
China
Email: kpchan@cs.hku.hk

T.Y. Chen
Faculty of Information and
Communication Technologies,
Swinburne University of Technology,
Hawthorn 3122,
Australia
Email: tychen@it.swin.edu.au

Dave Towey
Division of Science and Technology,
BNU–HKBU UIC,
Jinfeng Road,
Tangjiawan, Zhuhai,
Guangdong Province 519085, China
Email: dptowey@cs.hku.hk

Abstract— **Adaptive Random Testing (ART) methods are Software Testing methods which are based on Random Testing, but which use additional mechanisms to ensure more even and widespread distributions of test cases over an input domain. Restricted Random Testing (RRT) is a version of ART which uses exclusion regions and restriction of test case generation to outside these regions. RRT has been found to perform very well, but incurs some additional computational cost in its restriction of the input domain. This paper presents a method of reducing overheads called Forgetting, where the number of test cases used in the restriction algorithm can be limited, and thus the computational overheads reduced. The motivation for Forgetting comes from its importance as a human strategy for learning. Several implementations are presented and examined using simulations. The results are very encouraging.**

I. INTRODUCTION

In Software Testing, there has been much debate over the use of Random Testing (*RT*) as an effective method [9], [10], [15], [19]. When testing software, a *test case* is a combination of inputs to the software which represent a single use of the software; Random Testing refers to the selection of test cases at random from the input domain. There are many reasons why Random Testing remains a popular choice: in addition to its simplicity and the efficiency of test case generation [10], reliability estimates and statistical analyses are also easily performed. In fact, many real-life applications do make use of Random Testing [16], [17], [18], [20].

Failure-causing inputs are those test cases which, when applied to the software cause a failure or reveal a problem in the software. Chan et al. [2] observed that the performance of some testing strategies may be influenced by the pattern of the failure-causing inputs in the input domain (the failure pattern). This prompted investigation into improving *RT*'s performance by incorporating information about failure patterns. Methods based on Random Testing, but involving additional strategies to take advantage of failure pattern insights have been named Adaptive Random Testing (*ART*) methods [5], [7], [8], [12], [13].

An insight from the *ART* research was that more a widespread or even distribution of the test cases over the input domain was more favourable for failure-finding, in certain situations. The intuition underlying why this is so may be explained by means of a simple example, as follows: Consider a two dimensional input domain with a circular failure region, the centre of which is O, and the radius of which is R (Fig. 1) Suppose a test case t is randomly generated to test the program, but does not reveal a failure (that is, t falls outside the failure region) . Although both the location of O and the value of R are unknown, O is clearly at a distance of at least of R from t. Obviously, any test case drawn from the circular failure region is sufficient to show that the program is faulty, but for illustration of the intuition, assume that the testing objective is to select O as a test case. In view of the absence of the knowledge of R, intuitively speaking, it is better to choose a test case far away from t rather one close to t. Since the input domain is bounded, "far away" will effectively mean "widespread" or "evenly distributed".

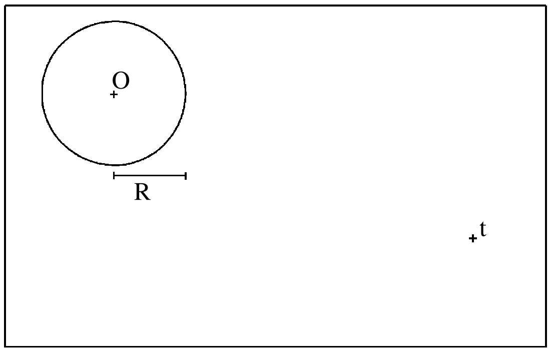

Fig. 1. Illustration of widespread distribution intuition

A version of *ART*, based on the use of exclusion, is the Restricted Random Testing (*RRT*) method [5]. 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.

One motivation behind the *RRT* methods is the intuition that, by incorporating additional information into the test case selection/generation process, it should be possible to improve the testing results [6]. This corresponds to a learning aspect of the method, and the results have shown the intuition to be correct – in experiments, the *RRT* method has outperformed *RT* by up to 80% on some occasions [5].

With the additional information come additional overheads. Previous research into *ART* has yielded Filtering [4] and Mirroring [3], [7] strategies, both of which reduce the computational costs associated with *RRT*. In this paper we present a new overhead reduction method, *Forgetting*. *Forgetting* is inspired partly by research into the importance of forgetting as a human strategy for learning, and application of this research in Machine Learning [14]. The rest of this paper is laid out as follows: In the next section, test case generation according to the basic *RRT* algorithm is outlined. In Section III the computational overheads of *RRT* are examined, and the aim of *Forgetting* explained. Section IV gives details about the motivation behind *Forgetting*. Some different implementations of *Forgetting* are explained in Section V, and the application of these implementations to some simulations is investigated in Section VI.

II. Restricted Random Testing - Method

When testing according to the *RRT* method, given a test case that has not revealed failure, rather than simply select another test case randomly, the area of the input domain from which subsequent test cases may be drawn is restricted. In two dimensions, a circular exclusion zone around each non-failure-causing input is created, and subsequent test cases are restricted to coming from outside of these regions. By employing a circular 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. For example, in two dimensions (2D), with a target exclusion region area of A, if there are n points around which we wish to generate exclusion zones, then each exclusion zone area will be A/n, and each exclusion zone radius will be $\sqrt{A/(n\pi)}$.

A graphical representation of the generation of the first few test cases, using the *RRT* method, is shown in Fig. 2. As shown, after each test case is generated (and presumably applied to the program without revealing failure), exclusion zones are created around all non-failure-causing test cases, and the next test case is selected from outside these excluded regions.

A feature of the *RRT* method, particularly for the circular exclusion region version [4], is that there is often a difference between the Target Exclusion Ratio and the Actual Exclusion [5]. As elsewhere, the Exclusion Ratios in this paper refer to Target Exclusion.

III. Overheads

The *RRT* methods incur potentially significant overheads in the generation of the $(m+1)^{th}$ test case. At this instant, when generating the $(m+1)^{th}$ test case, there are already m exclusion regions around m executed test cases, and the $(m+1)^{th}$ test case is restricted to coming from outside these regions. A simple implementation of the exclusion region is to ensure that the candidate test case is a greater distance from each executed test case than the radius of the exclusion region. For two points, P and Q ($(p_1, p_2, \ldots, p_N)$ and $(q_1, q_2, \ldots, q_N)$), the Euclidean distance between the points can be calculated from the following expression:

$$Distance(P,Q) = \sqrt{\sum_{i=1}^{N} (p_i - q_i)^2} \quad (1)$$

Ignoring possible optimizations, in a best case scenario, where the first candidate test case is outside all exclusion regions, there are m distance calculations required to confirm that the $(m+1)^{th}$ test case is acceptable. In practice, it is possible that several attempts at generating an acceptable test case will be required. For each unacceptable candidate, there will have been x number of comparisons (and hence x distance calculations) prior to that comparison revealing the test case to be within an exclusion region. The value of x will be between 1 and m, the worst case being that the candidate is found to be within the final exclusion region checked.

The *Forgetting* methods attempt to reduce the overheads by limiting the number m of exclusion regions, and hence the number of distance calculations required to verify whether or not the $(m+1)^{th}$ test case is acceptable (not within an exclusion region).

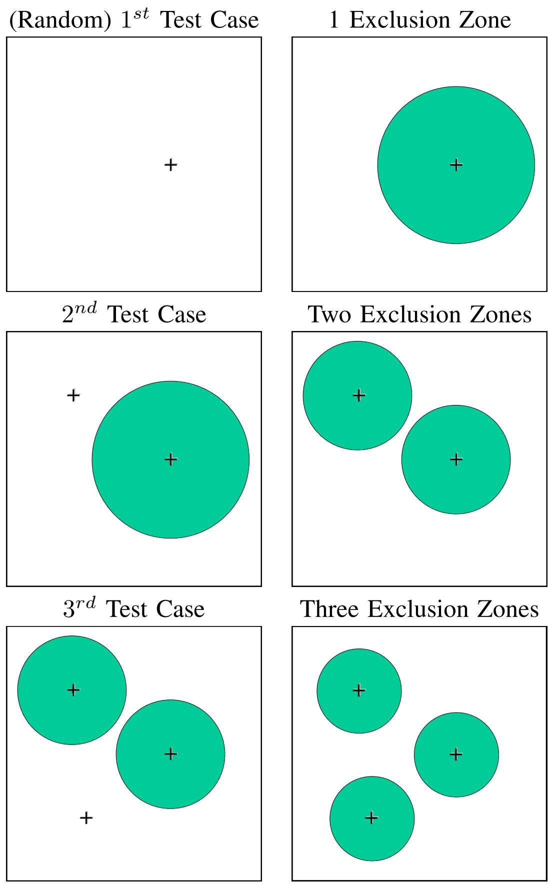

Fig. 2. Example of RRT Test Case generation process for first few test cases

IV. Motivation for Forgetting

Human learning is often characterised by inaccurate retention or recall, termed forgetting, which as Markovitech and Scott pointed out, "is usually regarded as an unfortunate failure of the memory system [11]." Markovitech and Scott explored the potential benefits of such failure in the context of Machine Learning [14], finding that in addition to the obvious reductions in overheads, even random deletion of knowledge yielded improvements in system performance.

For *RRT*, *Forgetting* is motivated mainly by the desire to reduce overheads. A feature of the *RRT* method is that as the number of executed test cases increases, the size of each individual exclusion zone decreases. This allows the selection of new test cases to be increasingly close to previously executed test cases, as we want, but it also increases the computational burden; the exclusion radius decreases in size such that at its extreme, when the radius is of negligible length, we are effectively performing Random Testing, but with considerably higher overheads. If *Forgetting* is applied in such a way that, for example, only a maximum of k test cases were used in the *RRT* algorithm, then the minimum size of exclusion zones would be known in advance, and some assurance of the cost-benefit trade-off would be available.

Chan et al. [1] previously examined the distribution of *F-measures* (numbers of test cases required to find a first failure) for the Distance-based implementation of *ART* (*DART*). Their analysis suggested that the advantage of continuing to run the algorithm after a certain number of test cases decreased, and that a reset/restart may be more appropriate. We conducted a similar examination of *RRT* by simulating a faulty program in two dimensions, with a failure rate (θ) of 1%, and applying *RRT* with an exclusion ratio of 150% (a value close to the *Max R*, the optimal value for the exclusion ratio [5]). The average *F-measure* was calculated over a sample of 5,000, and the distribution of the *F-measures* for various ranges calculated. In this simulation, the average *F-measure* was found to be 65.50, which is about a 35% improvement over the expected *F-measure* for *RT* (The expected *F-measure* by Random Testing is the inverse of the failure rate, so in the simulation it is 100). Table I summarizes the results, showing a skewed distribution of *F-measures* for *RRT*, suggesting that a reset/restart may also be appropriate for *RRT*.

TABLE I
Distribution of *F-measures* for RRT, mean = 65.50. Failure Rate = 1%, Expected *F-measure* for *RT* = 100

F-measure Range		**Samples within Range**	**Cumulative Total Samples**	**Probability of finding failure**
lower	**upper**			
0	10	443	443	0.0886
10	20	414	857	0.1714
20	30	438	1295	0.2590
30	40	451	1746	0.3492
40	50	418	2164	0.4328
50	60	439	2603	0.5206
60	70	386	2989	0.5978
70	80	379	3368	0.6736
80	90	311	3679	0.7358
90	100	274	3953	0.7906
100	110	233	4186	0.8372
110	120	195	4381	0.8762
120	130	157	4538	0.9076
130	140	102	4640	0.9280
140	150	98	4738	0.9476
150	160	74	4812	0.9624
160	170	60	4872	0.9744
170	180	40	4912	0.9824
180	190	27	4939	0.9878
190	200	19	4958	0.9916
200	210	12	4970	0.9940
210	220	8	4978	0.9956
220	230	4	4982	0.9964
230	240	6	4988	0.9976
240	250	5	4993	0.9986
250	260	3	4996	0.9992
260	270	0	4996	0.9992
270	280	0	4996	0.9992
280	290	1	4997	0.9994
290	300	2	4999	0.9998
300	310	1	5000	1.0000

V. Forgetting Test Cases

Three implementations of *Forgetting* were investigated: *Random Forgetting*; *Consecutive Retention*; and *Restarting*.

Random Forgetting refers to an implementation of *Forgetting* where, at the time of generating the $(m+1)^{th}$ test case, given a value k (the *Memory Parameter*), we delete $m-k$ executed test cases, and apply the *RRT* algorithm only on the remaining k. This is the simplest version of *Forgetting*, but is also the least likely to retain the failure-finding efficiency rates of the basic *RRT* method. Because of the relationship amongst consecutive, namely that test case $i+1$ will be at least a distance of r_i (the exclusion radius for i test cases) from test case i, random deletion may result in a spread of remaining test cases with less even or widespread distribution than would be desired.

The second *Forgetting* implementation, *Consecutive Retention*, again given a *Memory Parameter* of k, and generating the $(m+1)^{th}$ test case, will delete the first $m-k$ test cases, retaining the last, consecutive, k test cases. Because this version maintains the relationship among consecutive test cases, it is expected that it will outperform *Random Forgetting*. Additionally, to investigate the importance of the initial k test cases, i.e., those before the deletion of test cases begins, two versions of *Consecutive Retention* were examined: one where the normal *RRT* method was applied, resulting in a decreasing exclusion region size for each of the test cases 1 to k; and a second version which applies a fixed size exclusion zone appropriate for k test cases to all test cases, including 1 to k.

The third *Forgetting* implementation, *Restarting*, involves a complete reset of the algorithm. After k test cases, and exclusion regions implemented as usual, *Restarting* entirely *forgets* everything that has happened, and restarts the *RRT* method. This version is the most obvious implementation inspired by the analysis of *RRT F-measure* distribution (Section IV).

VI. Empirical Study

The effectiveness of the different methods of *Forgetting* was examined through their application to simulations. The first simulations were the same as those explained in Section IV, namely a square input domain with a randomly located failure region representing 1% of the entire input domain. For reference, additional experiments were carried out using a smaller (0.1%) failure rate. The expected *F-measure* by Random Testing is the inverse of the failure rate, so in the simulations it is 100 and 1,000 [5].

Each of the *Forgetting* methods (*Random Forgetting*, both versions of *Consecutive Retention*, and *Restarting*) was applied to the simulations with various values for the exclusion ratio (*R*), and for various values of the *Memory Parameter*. In each case, the simulation was repeated 5,000 times, and the average *F-measure* calculated.

Table II shows the generated *F-measure* results for when the *Random Forgetting* method was applied to the first simulation. The results for the two versions of *Consecutive Retention* (the first version using *RRT* for the initial k test cases, and the second version applying a fixed size exclusion region for all test cases) are given in Tables III and IV. Because the second version applies a fixed size exclusion region to all test cases, a different range of Target Exclusion Ratios was used. Table V shows the results for the *Restarting* method.

Table I showed the distribution of *F-measures* obtained by the original *RRT* method, with a Target Exclusion Ratio of 150%. Because the failure rate (θ) was 1%, the expected *F-measure* by Random Testing (*RT*) is 100. The *RRT* method averaged an *F-measure* of 65.50 in the simulation, a significant improvement over *RT*. The Target Exclusion Ratio of 150% was selected because it is known to be a good approximation of the *Max R*, the optimal value [5]. In fact, it was found that in the simulations, a slightly better average *F-measure* was obtained with a Target Exclusion Ratio of 140% (65.13), although the difference is not large. The results showed that, for the simulation, *RRT* was able to locate a failure region with less than 100 test cases (i.e., the *F-measure* was less than 100) in about 4,000 of the 5,000 trials. It was also the case that all 5,000 trials found failure regions with less than 310 test cases.

For the some of the *Forgetting* methods, when applied to this simulation, when the *Memory Parameter* is sufficiently large, it would be expected that very similar results to the basic *RRT* could be obtained. In particular, for the *Random Forgetting* and *Restarting* methods, and decreasing exclusion region version of *Consecutive Retention*, when the same Target Exclusion Ratio is used, the results should be the same as for the basic *RRT* method. This is the case for when the *Memory Parameter* (k) is 500, 750, and 1000. Even when the *Memory Parameter* is lower, but still close to 300, the results are very comparable to those obtained without *Forgetting*, as can be seen in Tables II, III, and V.

It has been observed that as the Target Exclusion Ratio (*R*) is increased, the failure-finding performance improves, usually with the best performance obtained when the *Max R* is used [5]. In this study, the best performance for *RRT* was found when *R* was 140% (*Max R* was 150%). With the exception of the fixed size exclusion region version of the *Consecutive Retention* method, the *Forgetting* methods also appeared to yield best results when *R* was 140%, and also displayed the characteristic improvement curve for increasing values of *R* up to *Max R*. This was not the case in two situations: for the *Consecutive Retention* method when the exclusion region size was fixed (i.e., did not decrease as the number of test cases increased); and for combinations of relatively low *Memory Parameter* values with relatively high values for *R*.

The fixed size exclusion region version of the *Consecutive Retention* method, because it applies an exclusion region size appropriate for k test cases (even when the total number of test cases is less than k), will not have a *Max R* in the same way the other methods do. It does, however, display the characteristic curve with improvement in *F-measure* as *R* increases. Although Table IV appears to indicate that the best results are obtained when the method is applied with *R* equal to 200%, and the *Memory Parameter* equal to 140 (yielding the *F-measure* of 66.29), similar, and possibly better results might be obtained with a higher values for both *R* and the *Memory Parameter*: Because the exclusion region size (and hence the exclusion radius) is fixed, the same size can be obtained by different combinations of the two parameters, e.g., *R* equal to 200%, and *Memory Parameter* equal to 200 will give the same size exclusion region as *R* equal to 100%, and *Memory Parameter* equal to 100; the only difference being that the former will retain 200 test cases for comparison, and should therefore have similar or better performance (which can be verified from Table IV).

The second situation where the simulation results were not as might be expected for *RRT* was for particular combinations of the *Memory Parameter* and *R*. As explained in Section IV, *RRT* allows selection of areas increasingly close to previously executed test cases by contracting the exclusion regions after each new test case is executed. Because there is a limit on the number of test cases used when applying the *Forgetting* algorithm, there is a limit on the contraction of the exclusion regions. If the limit is not sufficiently high for particular Target Exclusion Ratios, then there may actually be a decline in the failure-finding efficiency. This can be seen for the lower values of the *Memory Parameter* (k) for all the *Forgetting* methods.

All *Forgetting* methods do show the potential to achieve similar failure-finding efficiency to the basic *RRT* method, with identical efficiency possible when the *Memory Parameter* is sufficiently large. Even with relatively small values for the *Memory Parameter*, with the corresponding reduction in overheads, there is still comparable failure finding efficiency.

Tables VII, VI, VIII and IX summarize the results for a second set of simulations, with a smaller failure rate (0.1%). For these simulations, the expected *F-measure* by *RT* is 1,000, and the average *F-measure* for the basic *RRT* method (applied with a Target Exclusion Ratio of 150%) was 601.16. As in the earlier simulations, each of the figures in the tables and was averaged over a sample of 5,000 iterations.

TABLE II

F-measures FOR *Random Forgetting* METHOD, FAILURE RATE = 1%

	Target Exclusion Ratio														
k	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**	**110%**	**120%**	**130%**	**140%**	**150%**
10	95.70	94.37	94.69	96.95	96.62	99.95	101.57	100.41	101.09	103.55	106.33	108.69	111.17	109.74	116.75
20	95.05	92.53	91.04	92.22	91.11	92.20	94.89	95.87	95.85	99.28	101.03	102.25	106.72	104.93	106.39
30	94.18	91.33	90.03	89.59	88.51	88.76	88.46	88.73	92.16	92.36	95.08	95.97	98.25	98.71	101.13
40	94.51	90.71	87.77	87.73	86.09	85.47	85.17	88.99	87.08	87.47	91.27	91.66	92.04	92.43	95.57
50	93.92	90.31	88.18	86.81	84.26	84.50	83.67	83.57	84.97	85.73	85.73	86.42	86.89	87.71	91.17
60	94.25	89.81	88.06	84.86	84.31	82.93	80.72	81.10	81.91	80.90	83.43	84.10	82.85	84.13	84.95
70	94.07	90.17	86.84	84.31	83.04	81.50	80.44	80.87	80.17	79.79	80.69	80.33	81.36	79.61	81.40
80	93.85	89.38	86.38	83.15	81.62	80.77	79.29	78.57	77.91	78.61	79.35	78.11	78.19	78.02	78.34
90	93.88	89.72	86.04	82.98	80.87	80.36	77.66	77.35	76.73	76.17	76.17	76.72	74.40	75.21	76.77
100	93.73	89.43	85.75	82.81	80.60	79.51	76.83	76.60	75.48	75.51	75.67	74.30	73.23	72.58	73.83
110	93.58	89.05	85.67	82.48	80.65	78.97	75.44	75.43	74.69	75.34	75.01	73.09	71.55	71.31	71.35
120	93.57	88.93	85.38	82.02	79.65	78.53	75.69	75.08	73.47	74.04	74.39	71.61	71.14	70.08	70.58
130	93.46	88.89	85.08	81.67	79.22	77.85	75.18	74.35	72.87	73.09	73.00	71.14	70.13	67.95	69.41
140	93.40	88.72	84.98	81.60	78.89	77.54	75.05	73.82	72.06	72.54	72.62	70.19	69.20	67.71	67.88
150	93.27	88.65	84.53	81.33	78.88	77.08	74.25	73.40	72.09	71.86	71.71	69.62	68.24	67.45	67.54
200	93.18	88.59	84.21	80.60	78.02	76.08	73.06	72.00	70.34	70.53	69.95	67.84	66.63	65.57	65.71
250	93.04	88.25	84.03	80.16	77.44	75.53	72.68	71.33	69.83	69.92	69.42	67.52	66.44	65.16	65.55
300	92.96	88.15	83.81	80.03	77.19	75.39	72.36	71.29	69.69	69.79	69.38	67.40	66.29	65.13	65.50
500	92.85	87.98	83.60	79.81	76.94	75.21	72.27	71.22	69.63	69.75	69.31	67.41	66.26	65.13	65.50
750	92.83	87.96	83.57	79.78	76.94	75.20	72.27	71.21	69.63	69.75	69.31	67.41	66.26	65.13	65.50
1000	92.83	87.96	83.57	79.78	76.94	75.20	72.27	71.21	69.63	69.75	69.31	67.41	66.26	65.13	65.50

TABLE III

F-measures FOR *Consecutive Retention* METHOD (INITIAL k TEST CASES WITH DECREASING EXCLUSION REGION SIZES), FAILURE RATE = 1%

	Target Exclusion Ratio														
k	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**	**110%**	**120%**	**130%**	**140%**	**150%**
10	95.63	94.55	95.31	98.47	99.26	98.23	99.29	100.76	101.73	102.88	106.23	106.98	111.17	116.13	117.21
20	94.20	91.08	89.78	89.09	89.15	90.01	90.17	94.02	94.83	97.90	100.08	100.77	105.41	105.90	110.16
30	93.54	89.69	87.88	85.82	84.44	85.44	84.68	86.02	88.25	89.22	90.50	93.53	93.90	97.27	98.22
40	93.08	89.06	86.18	84.53	82.35	82.43	81.98	82.71	82.28	82.65	82.40	84.44	86.73	87.11	89.61
50	93.20	88.87	85.75	83.38	81.52	80.65	78.95	78.93	79.26	79.09	80.12	79.41	80.73	81.01	83.43
60	93.20	88.65	85.11	82.62	81.07	79.06	76.73	77.35	76.26	76.46	77.77	77.39	76.55	76.11	78.69
70	93.49	88.67	84.69	82.02	79.77	78.06	75.92	74.97	74.51	74.74	75.06	73.81	74.43	73.31	75.46
80	93.34	88.39	84.68	81.12	78.95	77.21	74.65	73.91	73.22	73.89	73.91	72.16	71.39	71.16	72.31
90	93.30	88.47	84.37	80.77	78.50	76.46	74.09	73.44	72.21	72.47	73.35	70.80	70.02	70.07	71.08
100	93.22	88.57	84.12	80.33	78.01	76.13	73.67	73.19	71.55	71.60	72.14	70.10	69.03	68.64	69.71
110	93.27	88.43	84.18	80.31	77.61	76.29	73.30	72.63	71.13	70.98	71.86	69.88	67.97	67.46	68.30
120	93.23	88.37	84.00	80.10	77.59	76.14	73.24	72.24	70.77	70.67	71.04	68.97	67.64	66.69	67.27
130	93.15	88.26	83.79	80.23	77.63	75.96	72.96	72.13	70.44	70.73	70.36	68.38	67.24	66.54	66.91
140	93.15	88.12	83.82	80.08	77.46	75.92	72.94	71.87	70.55	70.39	70.01	67.98	67.15	66.16	66.27
150	93.00	88.14	83.83	80.25	77.45	75.75	72.75	71.67	70.23	70.34	69.86	67.77	66.92	66.00	66.04
200	92.91	88.12	83.64	79.96	77.13	75.28	72.46	71.33	69.78	69.88	69.37	67.48	66.34	65.20	65.56
250	92.84	88.00	83.54	79.84	77.09	75.24	72.32	71.21	69.66	69.76	69.29	67.42	66.24	65.14	65.52
300	92.84	87.97	83.60	79.82	77.04	75.24	72.28	71.19	69.65	69.74	69.30	67.41	66.25	65.13	65.50
500	92.83	87.96	83.57	79.78	76.94	75.20	72.27	71.21	69.63	69.75	69.31	67.41	66.26	65.13	65.50
750	92.83	87.96	83.57	79.78	76.94	75.20	72.27	71.21	69.63	69.75	69.31	67.41	66.26	65.13	65.50
1000	92.83	87.96	83.57	79.78	76.94	75.20	72.27	71.21	69.63	69.75	69.31	67.41	66.26	65.13	65.50

As was the case for the simulations with a 1% failure rate, the results for the 0.1% failure rate show a basic trend of improving failure-finding efficiency as the Target Exclusion Ratio (*R*) was increased (when the *Memory Parameter*, k was sufficiently large).

In both sets of simulations, it appears that all *Forgetting* versions achieve similar results, improving as both *R* and the *Memory Parameter* increased. As noted in [1], the issue of what value to use for the *Memory Parameter* is difficult. There appears to be a correlation between relatively good results and values of the *Memory Parameter* corresponding approximately to the Expected *F-measure* for Random Testing (the inverse of the failure rate, θ). Of course, knowing the failure rate in advance of testing is not possible, but in cases where expectations or approximations are available, these may be used to help guide the choice.

TABLE IV

F-measures FOR *Consecutive Retention* METHOD (ALL TEST CASES WITH SAME EXCLUSION REGION SIZES), FAILURE RATE = 1%

	Target Exclusion Ratio															
k	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**	**150%**	**160%**	**170%**	**180%**	**190%**	**200%**
10	95.82	94.69	95.23	97.92	99.29	98.27	99.28	100.19	100.80	101.98	117.51	119.69	123.91	126.83	133.11	138.45
20	94.64	91.60	90.05	89.54	88.42	87.97	89.06	91.49	93.98	95.47	107.37	104.46	111.41	115.05	118.92	127.00
30	94.26	90.56	88.31	86.21	84.37	84.30	84.58	84.33	85.98	86.98	97.61	99.83	106.44	107.60	111.52	112.67
40	93.98	89.95	86.93	84.93	82.78	82.36	81.65	81.24	79.98	80.98	88.90	91.78	92.23	94.65	99.66	103.36
50	94.27	90.07	86.84	84.37	83.30	81.31	79.80	78.58	77.74	77.58	79.91	83.28	83.90	87.83	90.96	94.27
60	94.52	90.37	86.90	84.83	82.48	80.62	78.00	77.62	76.38	75.27	75.59	75.08	77.69	79.56	80.06	83.32
70	95.12	90.67	87.11	84.13	82.03	80.05	78.36	76.23	74.94	74.12	72.30	73.05	73.03	73.39	74.70	77.66
80	95.25	91.01	87.40	84.16	81.81	79.71	77.59	76.39	74.34	73.53	70.12	70.34	71.09	71.07	71.64	72.37
90	95.38	91.17	87.48	84.38	81.87	79.52	77.62	76.02	74.66	73.19	69.85	69.15	68.86	68.45	69.37	69.31
100	95.57	91.60	87.74	84.55	81.74	80.13	77.73	76.10	74.77	69.33	69.33	68.70	67.75	67.02	67.72	68.54
110	95.85	91.75	88.28	84.99	82.26	80.24	78.18	76.09	74.58	73.61	69.23	68.22	67.15	67.21	67.43	67.04
120	95.97	92.03	88.63	85.23	82.42	80.17	78.71	76.52	75.03	73.50	68.61	68.31	67.93	66.81	66.68	67.23
130	95.96	92.26	88.82	85.84	83.03	80.76	79.03	77.08	75.07	73.72	68.59	67.89	67.79	67.68	66.58	66.59
140	96.12	92.50	89.14	86.04	83.63	81.24	79.01	77.87	75.57	74.02	68.64	68.09	67.60	67.41	67.28	66.29
150	96.11	92.73	89.54	86.65	83.96	81.74	79.56	78.03	76.47	74.60	69.71	68.26	67.68	67.39	66.85	66.77
200	96.62	93.88	90.95	88.51	86.13	84.00	82.12	80.28	78.83	77.21	71.91	71.01	70.54	69.59	68.96	68.50
250	97.12	94.53	92.20	89.85	87.88	85.96	84.15	82.65	81.23	79.81	74.58	73.57	72.70	72.08	71.52	70.91
300	97.45	95.17	93.23	91.07	89.23	87.63	85.96	84.60	83.19	81.87	76.63	76.13	75.29	74.56	73.70	73.03
500	98.44	96.85	95.41	94.09	92.99	91.71	90.45	89.33	88.28	87.44	83.11	82.40	81.59	81.02	80.23	79.62
750	98.97	97.83	96.83	95.87	94.94	94.07	93.50	92.50	91.70	90.84	87.44	86.77	86.15	85.59	85.02	84.52
1000	99.27	98.42	97.56	96.83	96.06	95.39	94.76	94.07	93.65	92.98	89.94	89.32	88.84	88.28	87.88	87.44

TABLE V

F-measures FOR *Restarting* METHOD, FAILURE RATE = 1%

	Exclusion Ratio														
k	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**	**110%**	**120%**	**130%**	**140%**	**150%**
10	97.96	97.74	97.82	98.84	100.19	101.83	101.59	104.25	103.47	108.11	107.64	113.76	114.77	116.56	119.81
20	96.04	95.22	94.20	94.77	94.85	95.83	97.05	98.24	100.50	99.94	102.28	104.18	108.85	107.61	108.27
30	95.59	93.34	91.56	91.41	90.06	90.25	92.67	94.11	93.20	93.19	95.49	98.44	99.58	98.66	103.19
40	94.47	91.17	90.11	89.09	88.40	89.25	87.50	90.57	88.62	88.71	91.72	91.68	93.13	91.57	95.11
50	94.49	91.94	88.26	88.26	87.74	85.34	85.34	86.51	86.05	85.91	88.16	88.01	87.63	87.70	90.75
60	94.00	90.32	88.44	87.91	84.79	84.18	82.18	83.55	84.32	83.37	85.27	84.23	83.67	82.67	85.49
70	93.74	90.52	87.47	85.52	83.57	81.62	80.69	80.94	81.81	80.68	82.14	80.07	79.41	80.18	81.98
80	94.00	89.94	86.94	84.36	81.56	81.89	79.58	79.58	78.61	79.12	79.28	77.34	77.28	77.26	78.05
90	94.08	89.44	86.36	83.11	82.15	80.69	78.68	77.69	77.97	76.84	76.02	75.16	75.16	75.85	75.37
100	93.64	89.04	85.96	83.12	81.32	80.23	77.50	76.88	76.43	74.88	75.59	74.18	73.62	72.85	73.39
110	93.59	89.17	85.54	82.62	80.40	79.49	76.37	75.84	74.71	74.24	75.06	72.78	71.54	71.19	70.90
120	93.52	89.14	85.33	82.52	79.83	79.01	75.21	75.49	73.51	73.40	73.91	72.55	71.34	69.82	69.97
130	93.64	89.39	85.53	81.76	79.73	77.75	75.45	74.63	73.24	73.02	72.99	70.86	70.15	68.72	68.74
140	93.42	88.95	85.11	81.11	79.19	77.45	75.24	73.80	72.63	72.37	72.23	70.42	69.72	67.63	68.23
150	93.40	88.95	84.80	81.72	78.67	77.47	74.53	73.73	72.39	71.92	71.73	69.66	68.17	67.29	67.32
200	93.01	88.48	84.09	80.34	78.03	76.18	73.39	71.91	70.37	70.89	70.02	68.17	66.72	65.44	65.81
250	93.08	88.12	83.85	80.18	77.62	75.65	72.73	71.42	69.90	70.03	69.40	67.50	66.29	65.22	65.58
300	92.82	88.27	83.78	80.18	77.28	75.28	72.45	71.24	69.69	69.85	69.40	67.43	66.30	65.13	65.51
500	92.88	87.96	83.63	79.82	76.96	75.20	72.27	71.22	69.63	69.75	69.31	67.41	66.26	65.13	65.50
750	92.83	87.96	83.57	79.78	76.94	75.20	72.27	71.21	69.63	69.75	69.31	67.41	66.26	65.13	65.50
1000	92.83	87.96	83.57	79.78	76.94	75.20	72.27	71.21	69.63	69.75	69.31	67.41	66.26	65.13	65.50

TABLE VI

F-measures FOR *Random Forgetting* METHOD, FAILURE RATE = 0.1%

k	Target Exclusion Ratio			
	10%	**50%**	**100%**	**150%**
10	1011.60	1001.97	1062.66	1082.54
20	999.85	1000.24	1033.45	1030.33
30	990.82	994.62	1017.75	1059.02
40	978.96	977.79	1032.37	1059.86
50	982.29	1001.27	999.75	1054.97
100	966.09	943.93	1009.05	1037.50
150	960.29	911.52	961.57	1029.23
300	955.76	861.18	890.15	937.91
500	953.02	832.07	801.53	826.86
1000	946.33	794.57	706.35	668.98
1500	943.80	771.88	670.32	621.63
2000	941.24	762.70	657.97	604.26
4000	938.44	754.40	651.48	601.16

TABLE VII

F-measures FOR *Restarting* METHOD, FAILURE RATE = 0.1%

k	Target Exclusion Ratio			
	10%	**50%**	**100%**	**150%**
10	1002.20	1016.19	1036.80	1177.08
20	999.98	1023.64	1056.63	1140.15
30	1000.25	991.25	1036.25	1115.81
40	992.68	993.74	1054.80	1072.49
50	991.51	982.70	1045.17	1089.54
100	981.36	979.23	995.81	1020.07
150	966.31	947.35	959.08	989.93
300	956.81	895.06	902.57	884.91
500	951.13	844.05	805.22	808.22
1000	944.41	787.85	700.09	661.67
1500	939.83	779.62	670.24	613.95
2000	940.29	761.08	657.32	603.37
4000	938.33	754.56	651.48	601.155

TABLE VIII

F-measures FOR *Consecutive Retention* METHOD (INITIAL k TEST CASES WITH DECREASING EXCLUSION REGION SIZES), FAILURE RATE = 0.1%

k	Target Exclusion Ratio			
	10%	**50%**	**100%**	**150%**
10	999.55	1008.58	1045.02	1134.25
20	995.09	985.70	1030.50	1100.01
30	989.05	982.94	1012.56	1117.97
40	983.63	971.20	1018.27	1087.04
50	976.75	977.87	1007.05	1099.26
100	962.15	941.53	967.66	1061.13
150	952.05	901.36	921.53	1039.76
300	948.49	819.84	833.29	924.26
500	940.98	789.24	736.28	770.58
1000	939.62	763.97	668.14	629.06
1500	938.59	756.68	656.01	606.06
2000	937.46	755.14	652.22	602.05
4000	937.96	754.06	651.48	601.16

TABLE IX

F-measures FOR *Consecutive Retention* METHOD (ALL TEST CASES WITH SAME EXCLUSION REGION SIZES), FAILURE RATE = 0.1%

k	Target Exclusion Ratio					
	10%	**50%**	**100%**	**150%**	**175%**	**200%**
10	999.92	1007.04	1042.76	1138.21	1177.17	1351.74
20	995.71	984.80	1027.86	1125.74	1172.24	1247.25
30	988.39	980.00	991.97	1089.72	1163.53	1209.93
40	982.67	969.04	1008.95	1083.45	1117.09	1237.98
50	975.79	979.10	995.79	1084.63	1137.13	1238.72
100	962.99	935.86	961.26	1035.56	1122.05	1216.17
150	953.02	893.25	908.59	1010.58	1094.87	1185.05
300	952.78	819.94	812.94	923.17	981.32	1082.72
500	950.32	797.07	722.80	742.08	769.03	850.75
1000	959.35	815.21	705.45	660.00	642.73	633.86
1500	968.59	839.62	734.54	671.08	648.76	636.17
2000	975.31	861.61	766.63	704.80	682.72	662.71
4000	991.31	917.01	853.35	804.26	781.67	761.83

VII. SUMMARY

Restricted Random Testing (*RRT*) [5] is an implementation of Adaptive Random Testing (*ART*) [5], [7], [8], [12], [13] which uses exclusion regions and restriction of the Input Domain to achieve *ART*'s goal of a widespread and evenly distributed pattern of test cases.

Forgetting is a simple overhead reduction method which allows the algorithm to be applied to a limited number (the *Memory Parameter*, k) of previously executed test cases, thereby reducing computation costs. In this paper, we introduced four versions of *Forgetting* and applied them to some simulations previously tested using the basic *RRT* algorithm. Results showed that, when the *Memory Parameter* was sufficiently large, the *Forgetting* methods performed similarly to the basic *RRT* method, including yielding better results as the Target Exclusion Ratio increased. It was also noted that, if possible, the choice of value for the *Memory Parameter* may be guided by expected failure rates.

ACKNOWLEDGEMENTS

This project is partially supported by a Discovery Project grant from the Australian Research Council (DP0557246).

REFERENCES

[1] F. T. Chan, Kwok Ping Chan, T. Y. Chen, and Siu-Ming Yiu, "Adaptive Random Testing with CG Constraint", Proceedings *COMPSAC '04: Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)*, pp. 96-99, IEEE Computer Society, Washington, DC, USA, 2004.

[2] F. T. Chan, T. Y. Chen, I. K. Mak, and Y. T. Yu, "Proportional Sampling Strategy: Guidelines for Software Testing Practitioners", *Information and Software Technology*, Vol. 28, No. 12, pp. 775-782, December 1996.

[3] K. P. Chan, T. Y. Chen, Fei-Ching Kuo and Dave Towey, "A Revisit of Adaptive Random Testing by Restriction", Proceedings *COMPSAC '04: Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)*, pp. 78-85, IEEE Computer Society, Washington, DC, USA, 2004.

[4] K. P. Chan, T. Y. Chen, and Dave Towey, "Adaptive Random Testing with Filtering: An Overhead Reduction Technique", Proceedings *17th International Conference on Software Engineering and Knowledge Engineering (SEKE'05)*, Taipei, Taiwan, Republic of China, July 14-16, 2005.

[5] K. P. Chan, T. Y. Chen, and Dave Towey, “Restricted Random Testing: Adaptive Random Testing by Exclusion”, *International Journal of Software Engineering and Knowledge Engineering*, to appear, 2006.

[6] K. P. Chan, Dave Towey, T. Y. Chen, Fei-Ching Kuo and Robert Merkel “Using the Information: Incorporating Positive Feedback Information into the Testing Process”, STEP ’03: Proceedings of the Eleventh Annual International Workshop on Software Technology and Engineering Practice (STEP’03), pp. 71-76, IEEE Computer Society, Washington, DC, USA, 2003.

[7] T. Y. Chen, F. C. Kuo, R. G. Merkel, S. P. Ng, “Mirror Adaptive Random Testing”, *Information and Software Technology*, Vol. 46, No. 15, pp. 1001-1010, 2004.

[8] T. Y. Chen, H. Leung, and I. K. Mak, “Adaptive Random Testing”, *Lecture Notes in Computer Science*, Volume 3321, pp. 320-329, Jan 2004.

[9] R. Hamlet, *Random testing. Encyclopedia of Software Engineering. Edited by Marciniak, J.*, Wiley, 1994.

[10] P. S. Loo and W. K. Tsai, “Random Testing Revisited”, *Information and Software Technology*, Vol. 30, No. 9, pp. 402-417, September 1988.

[11] S. Markovitch and P. D. Scott, “The Role of Forgetting in Learning”, *Proceedings of the Fifth International Conference on Machine Learning*, Morgan Kaufmann, pp. 459-465, 1988.

[12] J. Mayer, “Adaptive Random Testing by Bisection with Restriction”, *Proceedings of the Seventh International Conference on Formal Engineering Methods (ICFEM 2005)*, LNCS 3785, Springer-Verlag, Berlin, pp. 251-263, 2005.

[13] J. Mayer, “Lattice-Based Adaptive Random Testing”, *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005)*, ACM, New York, pp. 333-336, 2005.

[14] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[15] G. J. Myers, *Art of Software Testing*, John Wiley & Sons, Inc., New York, 1979.

[16] B. P. Miller, L. Fredriksen, and B. So. “An Empirical Study of the Reliability of UNIX Utilities”. *Communications of the ACM*, Vol. 33, No. 12, pp. 32-44, December 1990.

[17] B. P. Miller, D. Koski, C. Lee, V. Maganty, R. Murthy, A. Natarajan, and J. Steidl. “Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services”. Technical Report CS-TR-1995-1268, University of Wisconsin, 1995.

[18] D. Slutz, “Massive Stochastic Testing of SQL”. *24th International Conference on Very Large Databases (VLDB 98)*, August 24-27, 1998, New York City, New York, USA, Proceedings, pp. 618-622, 1998.

[19] L. J. White, “Software testing and verification’, *Advances in Computers (26)*, pp. 335-391, 1987.

[20] T. Yoshikawa, K. Shimura, and T. Ozawa, “Random Program Generator for Java JIT Compiler Test System”. *3rd International Conference on Quality Software (QSIC 2003)*, pp. 20-24. IEEE Computer Society Press, 2003.