

## A SEGMENT BASED APPROACH FOR THE REDUCTION OF THE NUMBER OF TEST CASES FOR PERFORMANCE EVALUATION OF COMPONENTS

JOÃO W. CANGUSSU\*, KENDRA COOPER†  
and W. ERIC WONG‡

*Department of Computer Science, University of Texas at Dallas  
Richardson, TX 75080, USA*

*\*cangussu@utdallas.edu*

*†kcooper@utdallas.edu*

*‡ewong@utdallas.edu*

Component-based software development techniques are being adopted to rapidly deploy complex, high quality systems. One of its aspects is the selection of components that realize the specified requirements. In addition to the functional requirements, the selection must be done taking into account some non-functional requirements such as performance, reliability, and usability. Hence, data that characterize the non-functional behavior of the components is needed; a test set is needed to collect this data for each component under consideration. This set may be large, which results in a considerable increase in the cost of the development process. Here, a process is proposed to considerably reduce the number of test cases used in the performance evaluation of components. The process is based on sequential curve fittings from an incremental number of test cases until a minimal pre-specified residual error is achieved. The incremental selection of test cases is done in two different ways: randomly and adaptively. The accuracy and performance of the proposed approach are dependent on the values of the desired residual error. The smaller the residual error, the higher the accuracy. However, performance has an opposite behavior. The smaller the error, the larger the number of test cases needed. The results from experiments with image compression components are a clear indication that a reduction in the number of test cases can be achieved while maintaining reasonable accuracy when using the proposed approach.

*Keywords:* Component testing; performance testing; test case reduction.

### 1. Introduction

Component-based software engineering (CBSE) techniques hold the promise to support the timely, cost effective development of large-scale, complex systems; they are of keen interest to researchers and practitioners. CBSE methods propose to build software systems out of existing, re-usable components like using “lego” blocks. Since the components have already gone through the time consuming activities to specify, design, implement, test, and document their capabilities, the premise is that buying the components is a better decision, especially with

respect to development time and quality, in comparison to building the capabilities from scratch. However, there are numerous issues to address in CBSE including how to specify the functional and non-functional behavior of the components, how to evaluate, rank, and select components, how to predict the interoperability of components, etc.

The term component has a wide variety of definitions in the literature [5]. Here, a software component is an independent, reusable blackbox that, ideally, has a comprehensive interface description including:

- Specification of the functional and non-functional capabilities of the component. The non-functional description includes quality of service attributes (response time performance, memory, etc.)
- Programmer interface description, which defines how to use the component (e.g., an API definition with method names, parameter lists, return types, etc.)

A key issue in the specification of components is the problem of how to effectively test, or evaluate, the components in order to obtain quantitative data about their behavior. In particular, the non-functional behavior such as response time, performance, memory usage, etc. need to be collected. Recognizing that complete sets of test cases are not possible and large, comprehensive sets of test cases can be prohibitively expensive, techniques to reduce the number of test cases while still providing meaningful information about the components are needed.

Here, we present an approach to select a reduced set of test cases that can be applied to a wide variety of components and non-functional properties. The approach is based on the use of polynomial curve fitting techniques; the selection of an additional test case, which can be done either adaptively or randomly, is performed iteratively until an error tolerance is reached. The approach proposed here is an improvement over previous results [4]. While the previous approach was founded on a point based fitting, the improved approach relies on a segment based fitting. The experimental results show a considerable improvement not only in reducing the number of test cases but also on decreasing the residual error.

The approach is validated experimentally by selecting a reduced set of test cases for evaluating image compression components. The performance of the new approach shows a clear improvement when compared to prior results [4]. When comparing the random selection of an additional test case with the adaptive approach, our results favor the adaptive approach over the random; it produces a smaller number of test cases while still presenting reasonable accuracy.

The remainder of this paper is organized as follows. The new test case reduction approach is presented in Sec. 2; the evaluation of the new approach is in Sec. 3. An analysis of the impact of the value of the residual error is presented in Sec. 4. Related work is discussed in Sec. 5. Conclusions and future work are presented in Sec. 6.

## 2. Proposed Approach

The goal of the proposed approach is to reduce the number of test cases needed to do a performance evaluation of non-functional behaviors of components. In general, a large number of test cases are needed to obtain a precise evaluation. The conjecture here is that a reasonably accurate evaluation can be achieved with a considerable reduction in the number of test cases. Also, the approach needs to be general; it should not be restricted for use on a specific subset of non-functional attributes.

Any non-functional attribute of interest will always be a function of some input parameters, otherwise there is no need for testing. Therefore, the performance evaluation consists of finding the relationship between the input parameter(s) and the performance on the non-functional attribute. Both the input parameter (or some feature of the input parameter) and the non-functional attribute can be quantified and the relationship can be captured by some mathematical function. If the function is known in advance, then it can be directly used for the performance evaluation with no testing needed. However, this is rarely the case. In most scenarios, only the average performance is known which does not provide a comprehensive understanding of the behavior of non-functional attributes. In summary, the goal is to find the relationship using as few test cases as possible. Hereafter, the relationship between an input parameter  $x$  and a non-functional attribute  $y$  is referred to as  $y = f(x)$ .

Linear or non-linear regression models [19, 18] could be used to find the parameters of  $f(x)$  if the general format of the function is known. For example, if  $y$  is known to have an exponential behavior such as  $ae^{-bx}$ , regression models could be applied to find the values of  $a$  and  $b$  based on test cases (values of  $x$ ) and the observed performance (values of  $y$ ). However, in most cases, this relationship is not known and a more general approach needs to be used. Based on that, polynomial fit [11] has been chosen as, in general, any curve can be represented by a polynomial of a certain degree  $n$ ,  $p(x, n) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ . The problem is now the identification of the coefficients  $a_i$ ,  $i = 0, \dots, n$  of the polynomial. The *polyfit*( $x, y, n$ ) function available in MatLab [9] has been used to find the coefficients of  $p(x, n)$  that fits the data points  $(x_i, y_i)$ ,  $i = 1, \dots, m$  in a least square sense. In the case of performance evaluation,  $m$  is the number of test cases and the pair  $(x_i, y_i)$  represents the input value and the corresponding observed performance. The method of least squares is based on the minimization of errors (least square errors); the distance between the actual point and the point in the fitting curve. The best-fit curve of a given type is the curve that has the minimal sum of errors from a given data set. Suppose a sequence  $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$  of  $m$  data points are given. The fitting function  $f(x)$  has an error  $e$  associated with each data point, i.e.,  $e_1 = y_1 - f(x_1)$ ,  $e_2 = y_2 - f(x_2)$ ,  $\dots$ ,  $e_m = y_m - f(x_m)$ . Therefore, using the method of least squares, the best fitting curve has the property of minimizing the error  $\Pi$  as given by Eq. (1).

$$\Pi = d_1^2 + d_2^2 + \dots + d_n^2 = \sum_{i=1}^n d_i^2 = \sum_{i=1}^n [y_i - f(x_i)]^2. \quad (1)$$

The approach proposed here is an improvement over previous results. While the prior approach considers only the available points for the polynomial fit, the improved approach assumes that the points are connected by a line segment and the polynomial fit considers all the segments when doing the fit. For that reason, hereafter, we refer to the prior approach as *Test Case Reduction<sup>point-based</sup>* or simply *TCR<sup>pb</sup>*. The improved approach is referred hereafter as *Test Case Reduction<sup>segment-based</sup>* or simply *TCR<sup>sb</sup>*. The reasons to consider a segment based approach are exemplified in Fig. 1. It can clearly be seen that a segment based approach produces a better fit than a point based approach. Given any two consecutive points  $(x_i, y_i)$   $(x_{i+1}, y_{i+1})$ , a line is created connecting these two points and the fit is conducted based on the line and not only on the points. This process of populating the data with segments is referred in Figure 3-Line 06 as *FillLine(S)*. Given two consecutive points  $(x_i, y_i)$   $(x_{i+1}, y_{i+1})$  in the sequence S, *FillLine(S)* computes the line  $y = mx + b$  where the slope  $m = \frac{y_i - y_{i+1}}{x_i - x_{i+1}}$  and  $b = y_i - mx_i$ . Now, given a  $\Delta$  value, a sequence of  $x$  values  $x_i, x_i + 1\Delta, x_i + 2\Delta, \dots, x_i + k\Delta, x_{i+1}$  are created between  $x_i$  and  $x_{i+1}$ . These new  $x$  values are then used to compute the new  $y$  values using the line equation  $y = mx + b$ . The process is repeated for every two consecutive points in the sequence “S”.

Another aspect to be considered is the degree of the polynomial. That is, which degree should be used to fit the curve? Figure 2 shows the pseudo-code for the approach used here. Polynomial degrees ranging from 1 to 20 are used to do the fitting and the one producing the lowest root mean square error is selected. The upper

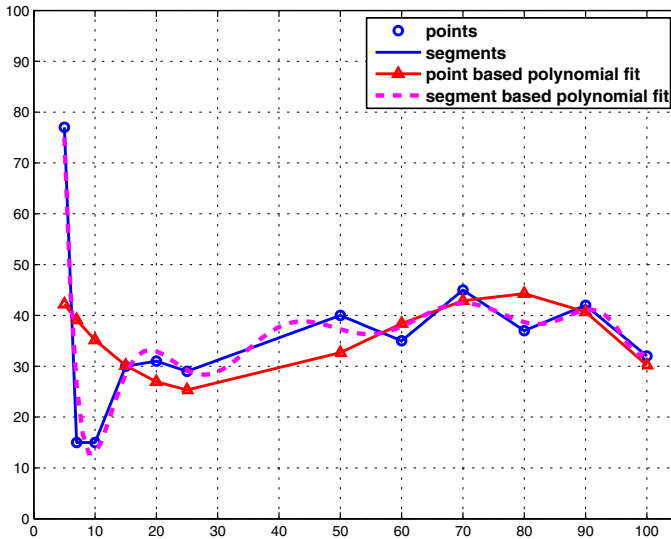


Fig. 1. Example of a segment based and point based polynomial fit for a given set of points  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ .

```

01 begin BestPolyFit(S)
02   residual_error = 10^10 ;
03   BestPoly = 0 ;
04   for pd = 1 to 20
05     // polyfit returns the coefficients of
06     // a polynomial of degree pd
07     P = polyfit(S,pd)
08     // polyval returns the polynomial values Y
09     // and the corresponding error ResEr
10     [Y, ResEr] = polyval(P,S)
11     if ResEr < residual_error
12       residual_error = ResEr
13       BestPoly = P
14     end
15   end
16   return BestPoly
17 end

```

Fig. 2. Polynomial degree selection algorithm for a given set of points represented by input S.

range of degree 20 has been selected since for all the experiments we have conducted the highest degree required was 12.

The pseudo-code in Fig. 3 presents the steps of the proposed approach ( $TCR^{sb}$ ). Let us assume that a performance evaluation needs to be done within a pre-specified range  $a, \dots, b$ . For example, if image compression components are being considered, one may be interested in images with size ranging from 1 K to 10 G bytes of memory. The first step is then to select an initial small set of test cases to start with. In our experiments (refer to Sec. 3) the starting testing suite  $T$  is composed of three test cases  $T = \{x_1 = a, x_2 = \frac{a+b}{2}, \text{ and } x_3 = b\}$ . The stopping criterion of the approach is based on the comparison of two consecutive fits. That is, the curve  $Fit_1$  is achieved using the results of test suite  $T_k = \{x_1, x_2, \dots, x_k\}$ , then a new curve  $Fit_2$  is computed using the test suite  $T_{k+1} = T_k \cup \{x_{k+1}\}$ , where  $x_{k+1}$  is a new selected test case. Now, the root mean square error (RMSE), as given by Eq. (2), between  $Fit_1$  and  $Fit_2$  is computed and the cycle stops when this error is less than a pre-specified threshold  $\epsilon$ . In this case, the last computed test suite is the one that can capture the main behavior of the performance of the non-functional attribute under consideration.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2}. \quad (2)$$

The behavior of the proposed solution is depicted in Fig. 4. Let us assume  $\epsilon = 10^{-2}$ . Fig. 4(a) shows the second iteration of the algorithm in Fig. 3. In this

```

01 begin
02   S = initial set of points and
03       associated outputs
04   error = infinity
05   while error > e
06     segS = FillLine(S)
07     Fit1 = BestPolyfit(segS)
08     S = S + new selected point
09     segS = FillLine(S)
10     Fit2 = BestPolyfit(segS)
11     error = RMSE(Fit1,Fit2)
12   end
13   TestSet = S
14 end

```

Fig. 3. Test composition algorithm

case, the first test suite  $T_5$  has four test cases and a new point (test case) marked with a square is selected to compose  $T_6$ . The value of the RMSE of the two fits is 0.9457 which is still larger than  $\epsilon$ . After one more iteration, as seen in Fig. 4(b), the RMSE between the fits using  $T_6$  and  $T_7$  results in an error of 0.0674 which means that more test cases are needed. The stopping criterion is reached in the next iteration (see Fig. 4(c)) where the error between  $T_7$  and  $T_8$  goes to  $0.0036 < \epsilon$ . In this case a total of eight test cases is needed to conduct the performance evaluation. As can be seen from Fig. 4(d), the curve computed with the results of only eight test cases is very similar to the actual curve as indicated by  $\text{RMSE} = 0.0254$ . That is, using only eight test cases the dominant behavior of the non-functional attribute has been properly captured.

In the example above, one aspect of the Algorithm in Fig. 3 has not been considered: how to select the new test case in Line 08? Two approaches are considered in this paper. The first simply randomly selects the new test case from the available test cases within the specified range. The second approach does the selection in an adaptive way. The larger the gap between two consecutive inputs (assuming the test cases have been sorted), the less information the fitting method has to cover that area. Therefore, the selection approach is to fill this gap and increase the information used by the fitting method.

The pseudo-code for the proposed adaptive selection algorithm is given in Fig. 5. After computing the gaps between each test case, the approach finds the largest gap (line 08) and then tries to find an available test case within this range. This step is needed because not all inputs in the original range from  $a$  to  $b$  may be available. For example, when testing the image compression with  $a = 1\text{ K}$  and  $b = 10\text{ G}$ , images from all these sizes may not be available. Very large images may be hard to find

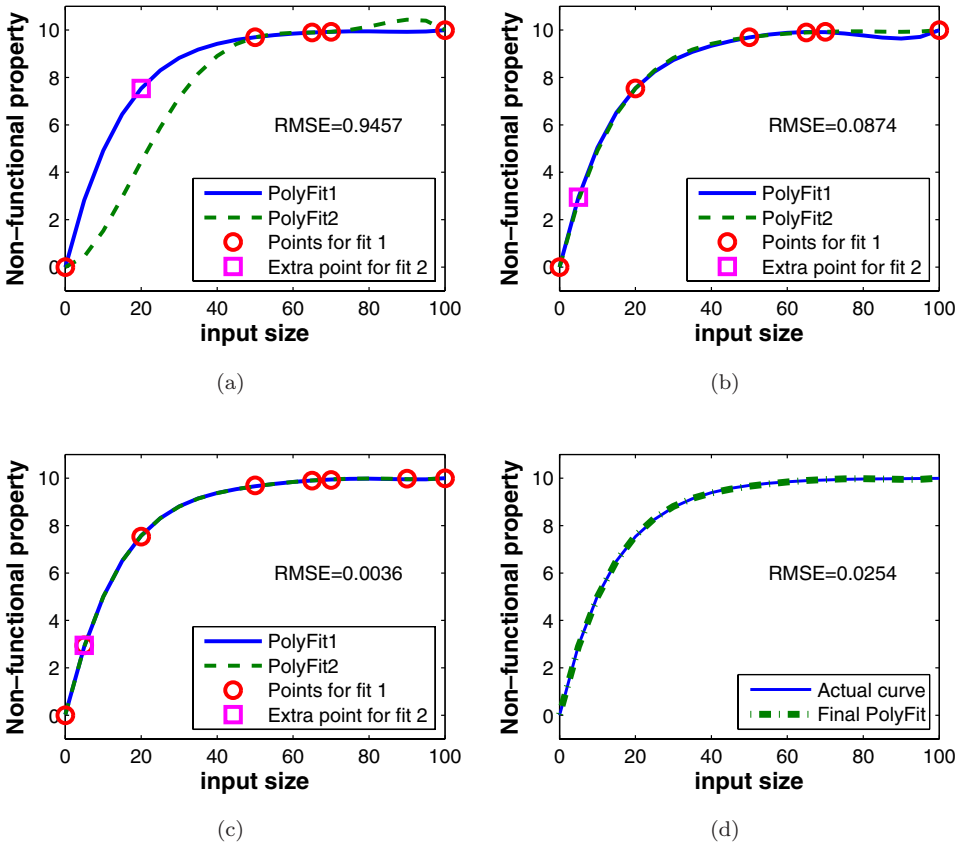


Fig. 4. Results from applying the algorithm from Fig. 3 to an exponentially shaped curve where (a) represents the results from iteration 2; (b) iteration 3; (c) iteration 4; and (d) is the comparison between the final curve and the actual one.

and only 3 images may be available in the range from 5 G to 10 G. If no input is available, then the algorithm searches for the next largest interval. Notice that this selection approach degrades into a full binary selection if all input values in the range are available; however, this is rarely the case.

### 3. Evaluation of the Proposed Approach

Three components for image compression are used here to evaluate the performance of the test case reduction technique described in Sec. 2. Although a large number of compression techniques exist, the decision to use Arithmetic Encoding (ARI) [20], Huffman coding (HUF) [1], and Burrows-Wheeler Transform (BWT) [3] is based on their generality and availability. Also, two non-functional attributes are used in the examples: compression time and compression ratio. These two have been selected to represent a linear and a non-linear non-functional attribute.

```

01 begin
02   T={x1,x2,...,xn} \\ T is sorted
03   for i=2 to n
04     Gap(i) = x(i) - x(i-1)
05   end
06   b = true
07   while b
08     max_interval = Max(Gap)
09     tc = FindInput(max_interval)
10     if tc == 0 \\ No input found
11       Gap = Gap - max_interval
12     else
13       b = false
14     end
15   end
16   x(n+1) = tc
17 end

```

Fig. 5. Adaptive test case selection algorithm.

A large set of 190 images is available for the performance evaluation. The images range in size from 10 to 10 M bytes. The images are not uniformly distributed. The non-uniform distribution is more realistic as smaller images (less than 1 M byte) are more common and easier to find than larger images (more than 5 M bytes). One image is used for each test case; the goal is to use the least number of test cases possible while still capturing the dominant behavior of the non-functional attribute.

Figures 6, 7 and 8 shows the results of applying the adaptive approach (algorithm from Fig. 5) to the evaluation of compression time for the three components. An error of  $\epsilon = 10^1$  has been used. Although the error may appear to be large at first, it is indeed small when compared to the values of the  $y$  axis ( $10^6$  milli-seconds). As we can see in Figs. 6, 7 and 8, the adaptive approach requires a total of 7, 28, and 8 test cases to capture the behavior of compression time for Huffman, BWT, and Arithmetic encoding, respectively. The observed behavior is linear and in the best case scenario only three points (test cases) would be needed to capture the behavior. However, the actual results are not a straight line and the use of only three points could lead to a different slope and possibly to a wrong characterization of compression time. In any case, the number of test cases needed seems reasonably small while appropriate to capture the behavior.

When comparing the results of  $TCR^{sb}$  and  $TCR^{pb}$ , presented in Table 1, for the adaptive approach we can easily see that the first outperforms the second in most cases. Only for BWT algorithm the required number of test cases has increased

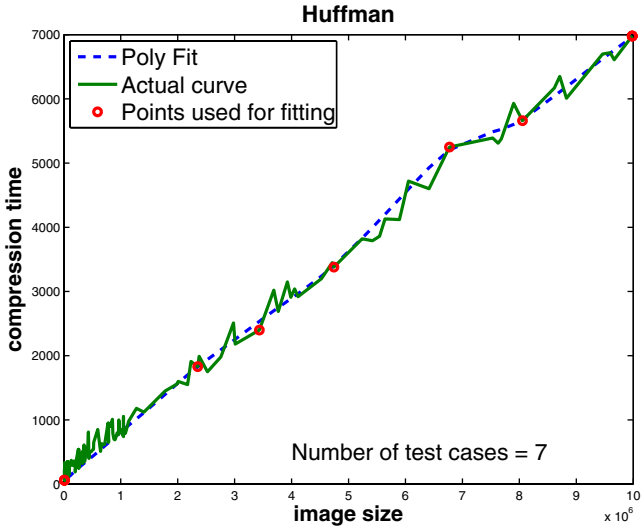


Fig. 6. Results of the adaptive selection of test cases for compression time using Huffman algorithm.

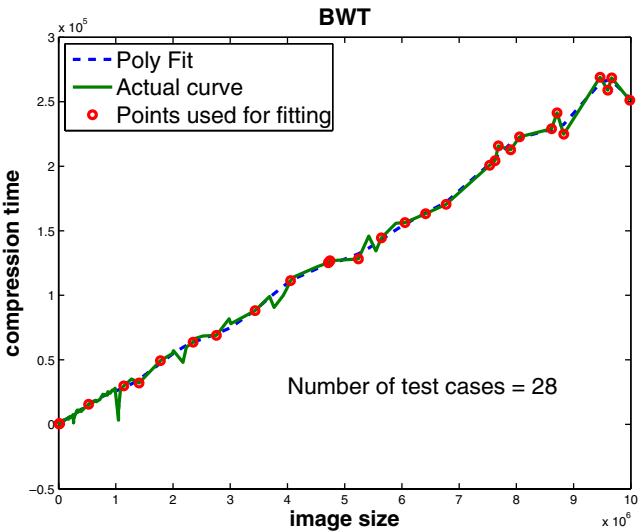


Fig. 7. Results of the adaptive selection of test cases for compression time using BWT algorithm.

by 21%; in the other two cases the decrease is substantial: 47% decrease for Arithmetic encoding and 30% for Huffman. On average, an 18% decrease on the number of test cases is observed. A similar behavior is observed for RMSE. A small increase of 3% is presented by the Huffman algorithm while the other two present a decrease of 29% (BWT) and 32% (ARI). An average decrease of 19% is observed

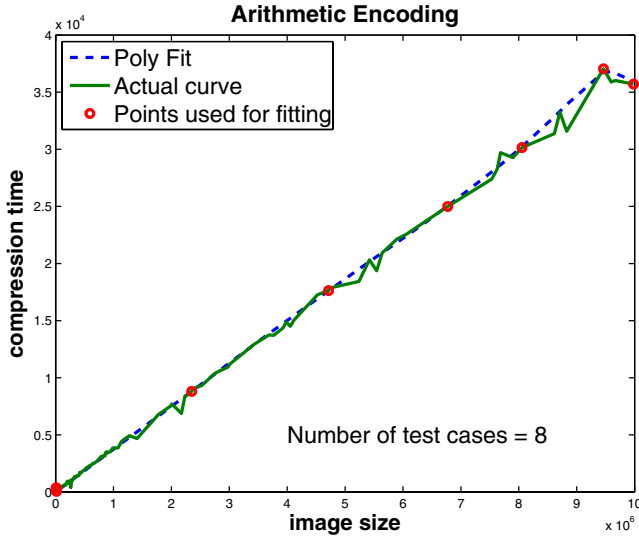


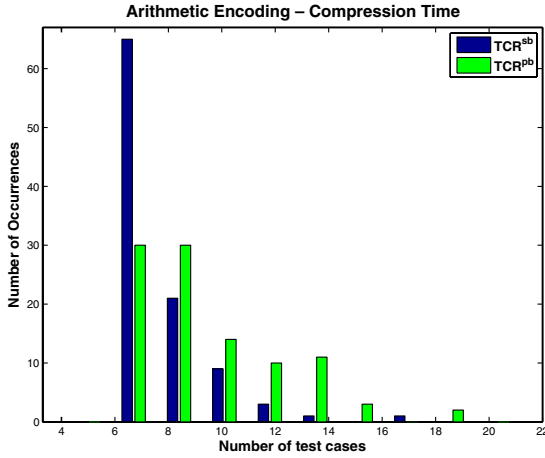
Fig. 8. Results of the adaptive selection of test cases for compression time using Arithmetic encoding algorithm.

Table 1. Results of the adaptive test selection approach for both  $TCR^{sb}$  and  $TCR^{pb}$  test case reduction approaches for compression time.

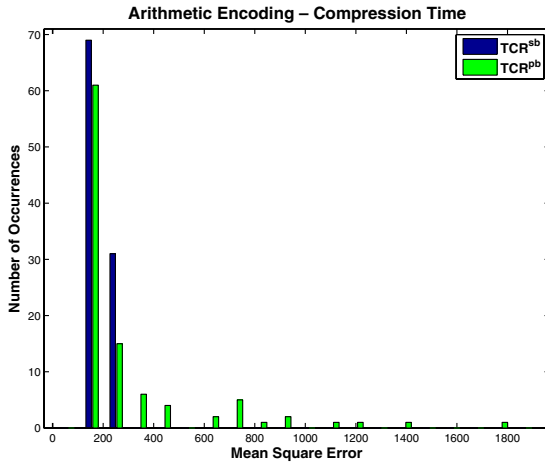
Adaptive Selection				
Approach	Compression Time			
	Test Cases		RMSE	
	$TCR^{sb}$	$TCR^{pb}$	$TCR^{sb}$	$TCR^{pb}$
Arithmetic Enc.	8	15	162.25	236.54
BWT	28	23	2648.90	3714.33
Huffman	7	10	115.09	111.29

on the residual error. The percentages are computed using the following equation  $\frac{abs(TCN_{TCR^{sb}} - TCN_{TCR^{pb}}) \times 100}{TCN_{TCR^{pb}}}$ ; where  $TCN_{TCR^{sb}}$  is the number of test cases for the segment based approach and  $TCN_{TCR^{pb}}$  is the number of test cases for the point-based approach.

The adaptive approach is deterministic; for a given set of images, it always produces the same sequence to be used as test cases in the performance evaluation. That is, the results for Figs. 6, 7 and 8 are always the same when the same set of images is used for the selection. The random selection performs differently whenever executed with a different seed. To better capture the performance of the random selection, a total of 100 simulation runs have been executed for each scenario. The



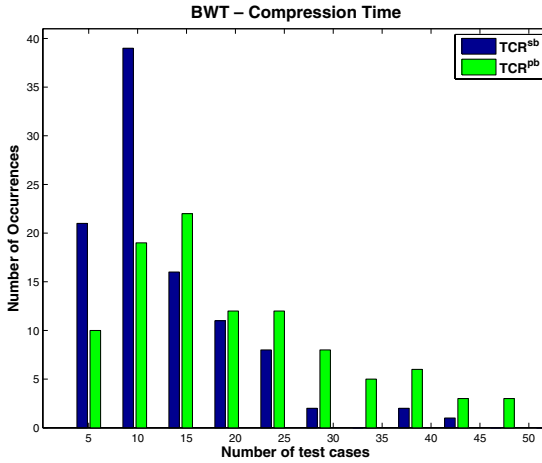
(a) Number of required test cases



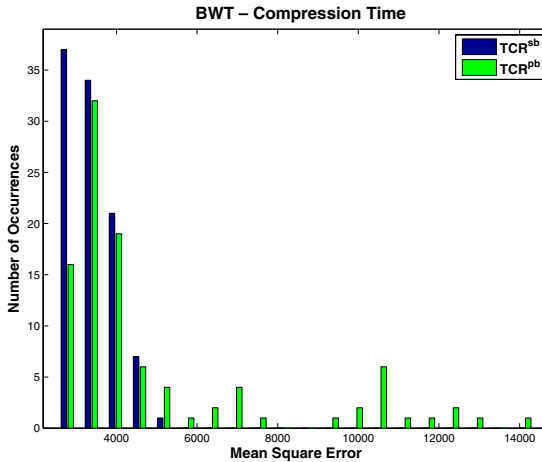
(b) RMSE (Root Mean Square Error)

Fig. 9. Results of 100 simulation runs for the random selection of test cases for compression time using Arithmetic encoding algorithm. The histograms show the results of both  $TCR^{pb}$  and  $TCR^{sb}$  approaches.

number of test cases needed for each of the three components and the respective frequency (number of executions that needed the same number of test cases) have been collected. The results for the three compression algorithms can be seen in Figs. 9, 10 and 11; the figures show the results for  $TCR^{sb}$  and  $TCR^{pb}$  for the random selection of test cases. As it can be seen, the new approach  $TCR^{sb}$  outperforms the  $TCR^{pb}$  approach in terms of the required number of test cases for all three algorithms. On average, an improvement (decrease in the number of test cases) of 30% has been achieved. The improvements for RMSE are even better with an



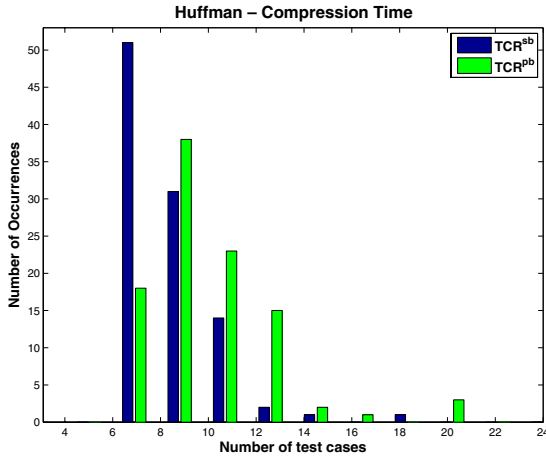
(a) Number of required test cases



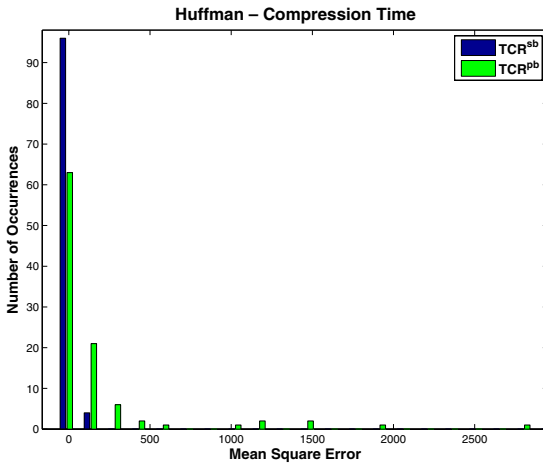
(b) RMSE (Root Mean Square Error)

Fig. 10. Results of 100 simulation runs for the random selection of test cases for compression time using BWT algorithm. The histograms show the results of both  $TCR^{pb}$  and  $TCR^{sb}$  approaches.

average decrease of 41%. Another aspect to be considered is the consistency of the new approach  $TCR^{sb}$ . Not only has the average improved, but standard deviations have decreased considerably. It can be seen that most results of the simulation runs for  $TCR^{sb}$  are concentrated on the left side while the results for  $TCR^{pb}$  are spread out. In addition, the number of outliers has presented considerable improvement mainly with respect to RMSE. Figures 9(b), 10(b) and 11(b) show that the results of  $TCR^{sb}$  are all concentrated within a small range which is not the case for  $TCR^{pb}$ . For example, Fig. 9(b) shows all the results for  $TCR^{sb}$  within a range from 0 to 250.



(a) Number of required test cases



(b) RMSE (Root Mean Square Error)

Fig. 11. Results of 100 simulation runs for the random selection of test cases for compression time using Huffman algorithm. The histograms show the results of both  $TCR^{pb}$  and  $TCR^{sb}$  approaches.

Most of the results for  $TCR^{pb}$  are within a 0 to 500 range but there are 8 simulation results outside this range (outliers). The average results for the simulation runs are presented in Table 2.

The results of using  $TCR^{sb}$  show some advantages, i.e. a smaller number of test cases, for the random selection over the adaptive selection. For the Arithmetic encoding, the results are very similar: 8 test cases are required by the adaptive and an average of 8.53 for the random. A small advantage is also observed for random approach for the Huffman algorithm. In the case of BWT the improvement

Table 2. Average results for 100 simulation runs of the random test selection approach for both  $TCR^{sb}$  and  $TCR^{pb}$  test case reduction approaches for compression time.

Approach	Random Selection			
	Compression Time			
	Test Cases		RMSE	
	$TCR^{sb}$	$TCR^{pb}$	$TCR^{sb}$	$TCR^{pb}$
Arithmetic Enc.	8.53	16.35	235.54	349.09
BWT	15.30	21.90	3675.37	5269.05
Huffman	8.93	10.73	103.94	255.35

is considerably large: a 46% decrease in the number of test cases is presented by the random approach. However, it should be noticed that the results for BWT present more variance than the other two. Regarding RMSE, the adaptive approach outperforms the random approach in all the cases. Which approach is recommended: adaptive or random? Only for BWT the random approach produces significantly better results for the number of test cases, but in this case there is a large variance and the chances of deviating from the average are considerably high. Combining this argument with the fact that the adaptive approach is more accurate (lower RMSE) lead us to conclude that the adaptive approach is a better alternative for test case reduction.

As stated before, compression time for the evaluated components presents a linear behavior with respect to image size. To further evaluate the proposed approach; compression ratio, a non-functional attribute with a non-linear behavior is analyzed next. In this case an error  $\epsilon = 10^{-2}$  have been used. The error is smaller because the scale for the  $y$ -axis (compression ratio) is comparatively smaller. Figures 12, 13 and 14 present the results from the adaptive selection. As it can be seen in the figures, the adaptive selection using  $TCR^{sb}$  required 8, 22, and 8 test cases for the three compression algorithms. Table 3 shows the results for both  $TCR^{sb}$  and  $TCR^{pb}$  approaches.  $TCR^{sb}$  presents better results than  $TCR^{pb}$  in all the cases with respect to the number of test cases. On average, the required number of test cases has been reduced by 23%. An opposite behavior is observed for RMSE; in this case  $TCR^{pb}$  outperforms  $TCR^{sb}$  by 66%. This behavior is expected as the initial points (small images) present a much higher compression ratio than large images. While  $TCR^{sb}$  adheres more closely to all the points,  $TCR^{pb}$  averages out the RMSE. This can be seen when comparing Fig. 7 to Fig. 13. The first is based on  $TCR^{sb}$  and the second on  $TCR^{pb}$ . The same is true for the other two compression algorithms.

With respect to the random selection for compression ratio the execution runs in Figs. 16(a), 17(a) and 18(a) present an average of 16, 24, and 17 test cases required to achieve the desired residual error when  $TCR^{sb}$  is used. This presents an average improvement of 69% when compared to the results using  $TCR^{pb}$ . Similar to compression time, the results in this case are more favorable to the adaptive

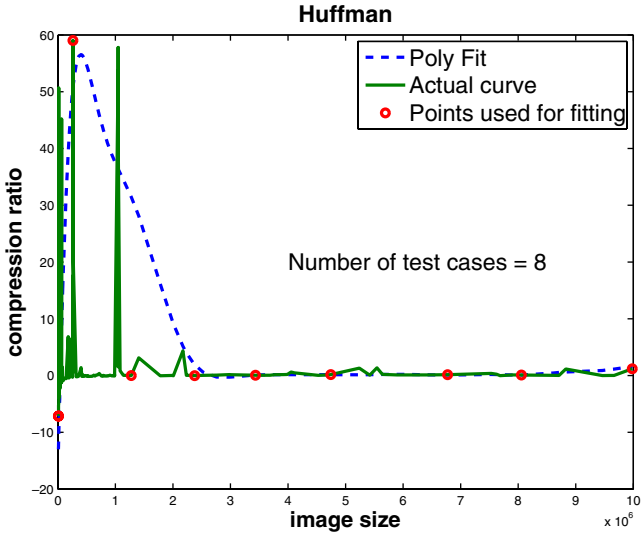


Fig. 12. Results of the adaptive selection of test cases for compression ratio using Huffman algorithm.

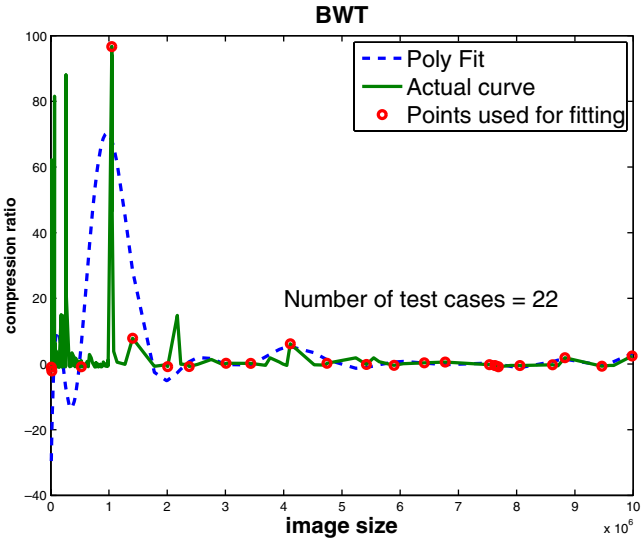


Fig. 13. Results of the adaptive selection of test cases for compression ratio using BWT algorithm.

selection than to the random selection of test cases. Adaptive selection using  $TCR^{sb}$  performs two times better than random selection for the Huffman component. The improvements for BWT and Arithmetic encoding are, respectively, 1.09 and 2.12. The average number of test cases needed for 100 simulation runs for the random

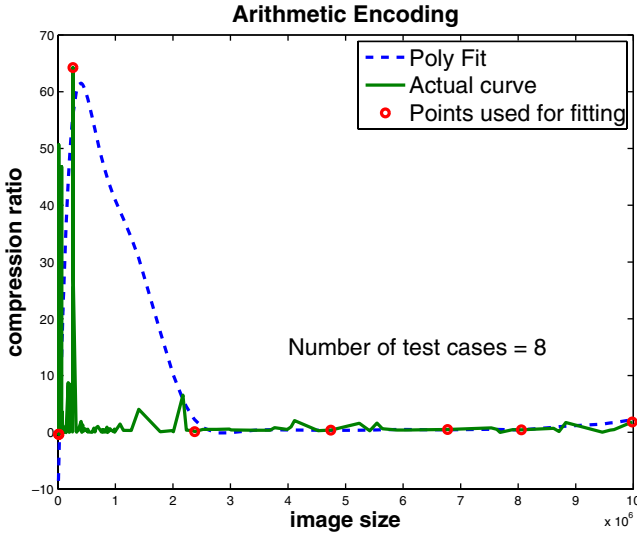


Fig. 14. Results of the adaptive selection of test cases for compression ratio using Arithmetic encoding algorithm.

Table 3. Results of the adaptive test selection approach for both  $TCR^{sb}$  and  $TCR^{pb}$  test case reduction approaches for compression ratio.

Approach	Adaptive Selection			
	Compression Ratio			
	Test Cases		RMSE	
	$TCR^{sb}$	$TCR^{pb}$	$TCR^{sb}$	$TCR^{pb}$
Arithmetic Enc.	8	11	21.41	11.71
BWT	22	24	23.02	16.37
Huffman	8	12	19.66	11.00

selection of test cases are presented in Table 4. Another point to be considered is that  $TCR^{sb}$  presents lower variance for the simulation runs than  $TCR^{pb}$  with respect to the number of test cases.

The results for compression ratio with respect to RMSE using a random approach present an opposite behavior than with respect to the number of test cases. Similar to the adaptive approach  $TCR^{pb}$  outperforms  $TCR^{sb}$  but in this case the improvement is not as significant. On average,  $TCR^{pb}$  outperforms  $TCR^{sb}$  by 10%. When comparing the random with the adaptive approach, the first outperforms the second by 43% on average.

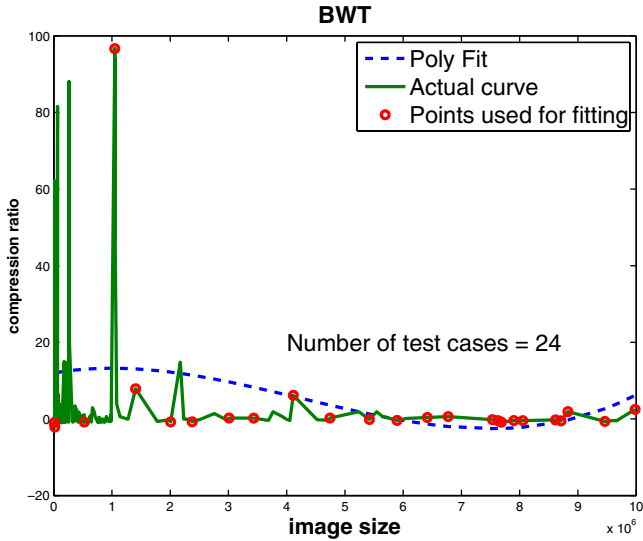
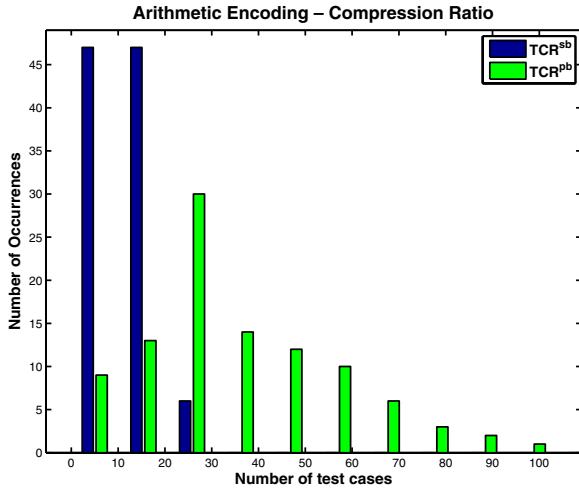


Fig. 15. Results of the adaptive selection of test cases for compression ratio using Huffman algorithm when  $TCR^{pb}$  is used.

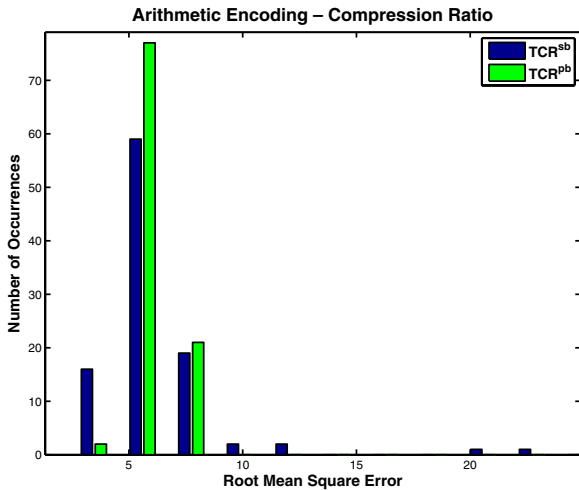
Table 4. Results of the adaptive and random test selection approach for both  $TCR^{sb}$  and  $TCR^{pb}$  test case reduction approaches for compression ratio.

Approach	Random Selection			
	Compression Ratio			
	Test Cases		RMSE	
	$TCR^{sb}$	$TCR^{pb}$	$TCR^{sb}$	$TCR^{pb}$
Arithmetic Enc.	16.10	40.91	7.20	7.08
BWT	24.46	93.39	19.08	17.83
Huffman	17.30	61.99	12.86	10.44

When considering compression ratio, the adaptive approach has performed slightly better than the random approach. This is due to the non-uniform distribution of the image sizes and the non-linearity of the requirement under consideration. In general, when test cases are uniformly distributed, the adaptive and the random selection approaches behave in a similar manner. The adaptive approach tries to fill the largest interval between two input values, which tends to make the selected inputs uniformly distributed. Therefore, if the inputs are already uniformly distributed, then the random selection is expected to behave in a similar manner. However, this is not the case for the compression ratio and since images are clustered, the selection of a new image may lead to a large difference between the two polynomial fits and consequently a larger number of test cases is required. The



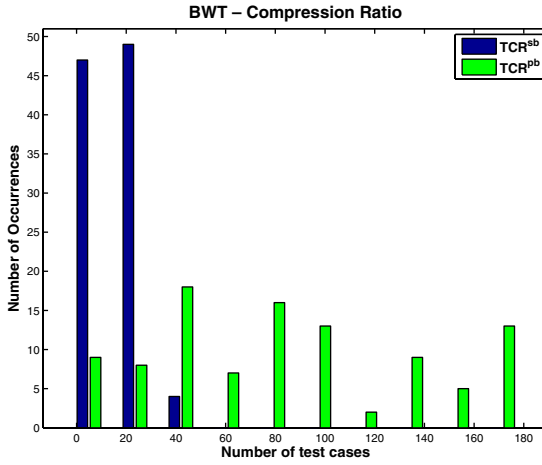
(a) Number of required test cases



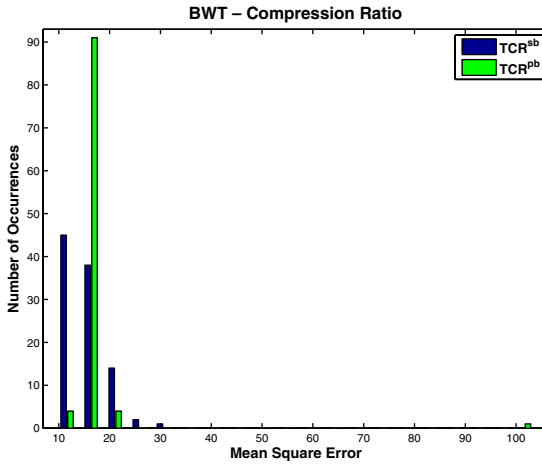
(b) RMSE (Root Mean Square Error)

Fig. 16. Results of 100 simulation runs for the random selection of test cases for compression ratio using Arithmetic encoding algorithm. The histograms show the results of both  $TCR^{pb}$  and  $TCR^{sb}$  approaches.

expectation is that the more complex the behavior of the requirement (meaning, the more complex the curve to fit), the larger the number of test cases (points) needed to capture its behavior. The non-linearity of compression ratio is an indication of this complexity leading to a larger number of test cases required to capture its behavior.



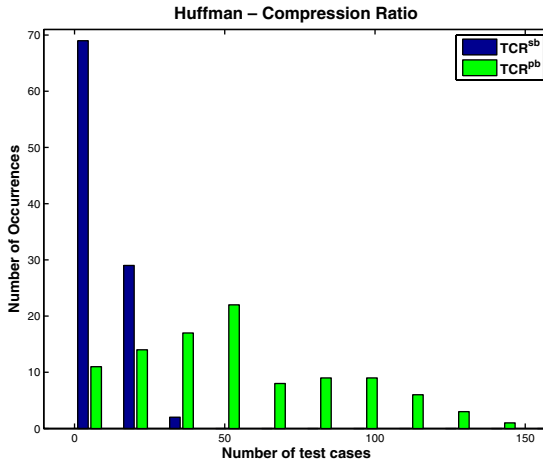
(a) Number of required test cases



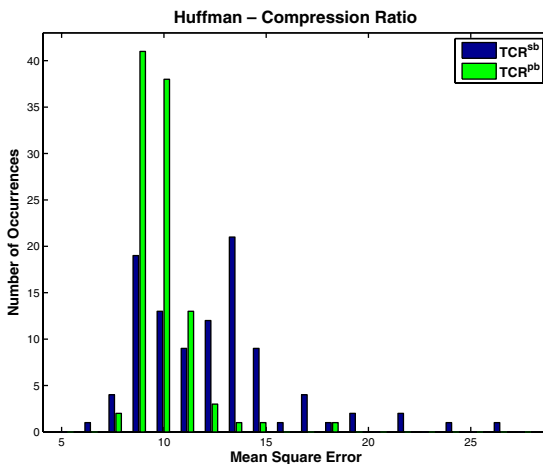
(b) RMSE (Root Mean Square Error)

Fig. 17. Results of 100 simulation runs for the random selection of test cases for compression ratio using BWT algorithm. The histograms show the results of both  $TCR^{pb}$  and  $TCR^{sb}$  approaches.

The results presented in this section provide a good indication that the new improved test reduction approach ( $TCR^{sb}$ ) can select a small number of test cases to conduct an accurate performance evaluation of components. However, one aspect that still needs an evaluation is the impact of the values of the error  $\epsilon$  on the performance of the approach. The conjecture is that the higher the degree of the polynomial and the smaller the value of  $\epsilon$  the larger the number of test cases required to capture the behavior. This issue is addressed in Sec. 4.



(a) Number of required test cases



(b) RMSE (Root Mean Square Error)

Fig. 18. Results of 100 simulation runs for the random selection of test cases for compression ratio using Huffman algorithm. The histograms show the results of both  $TCR^{pb}$  and  $TCR^{sb}$  approaches.

#### 4. Residual Error Impact Analysis

The only parameter that needs to be defined by the user when applying the proposed approach is the desired residual error  $\epsilon$ . Remember that the degree of the polynomial is automatically selected based on the minimization of the RMSE. A point of interest is the sensitivity of  $\epsilon$ . That is, what is the impact on the approach as  $\epsilon$  changes? Figs. 19 and 20 show the results when applying  $TCR^{sb}$  for compression time and ratio. In the first case, the values of  $\epsilon$  range from  $10^{-3}$  to  $10^3$  and in the second case the range is from  $10^{-5}$  to  $10^0$ . The ranges have been defined empirically

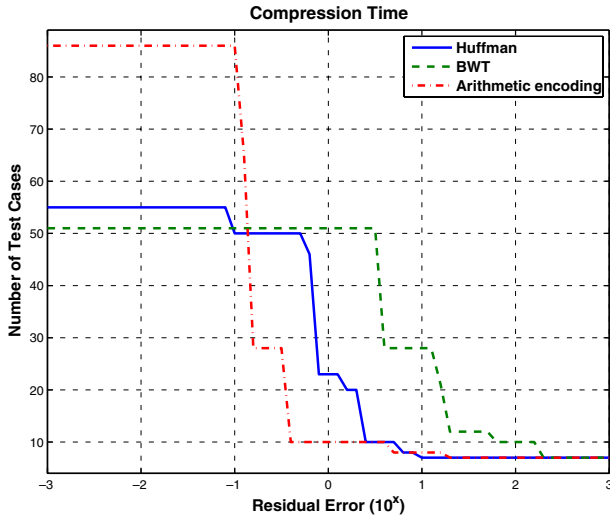


Fig. 19. Test cases required using the segment based adaptive approach ( $TCR^{sb}$ ) for the evaluation of compression time for residual errors ranging from  $10^{-3}$  to  $10^3$ .

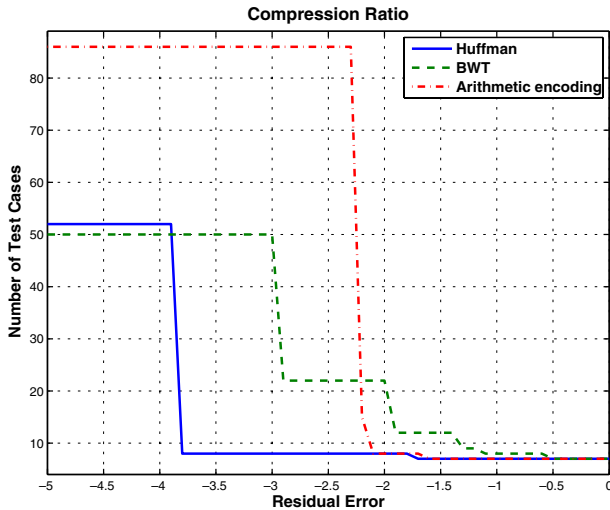


Fig. 20. Test cases required using the segment based adaptive approach ( $TCR^{sb}$ ) for the evaluation of compression ratio for residual errors ranging from  $10^{-x}$  to  $10^y$ .

so as to achieve an upper and lower bound in terms of the number of test cases required.

As expected, the smaller the value of  $\epsilon$  the larger the number of test cases required. That is, the larger the number of points needed to achieve the desired residual error  $\epsilon$ . This fact is confirmed by both Figs. 19 and 20. The interesting

aspect in this case lies more on the format of the results than on the actual values. That is, one would expect a more smooth decay in the number of test cases as the value of  $\epsilon$  increases. However, this is not observed in any of the cases. Notice that  $\epsilon$  changes in small intervals from  $10^i$  to  $10^{i+0.1}$ . Even in this case, the decay in the number of test cases is very sharp. Compression time (Fig. 19) presents the decay in a few steps but the decay for compression ratio (Fig. 20) happens in just one step for Huffman and Arithmetic encoding and three major steps for BWT.

The sudden drop in the number of test cases would be a problem if the drop occurs with very small  $\epsilon$  values. However, the drop occurs between  $10^{-1}$  and  $10^1$  for compression time which indicates an  $\epsilon$  that is at least  $10^5$  times smaller than the values under consideration. The drop for compression ratio occurs between  $10^{-4}$  and  $10^{-2}$  which also represents a small ratio for  $\epsilon$ ; at least  $10^4$  times smaller than the values under consideration. This is an indication that selecting a value of  $\epsilon$  that is around  $10^5$  times smaller than the values of the performance parameter under consideration would lead to a reasonably accurate approximation with a considerably small number of test cases.

## 5. Related Work

To our best knowledge, we are not aware of any studies with a similar objective as what is reported here. The rest of this section focuses on work in three categories: test cost reduction, adaptive testing, and component-based testing.

*Test Cost Reduction:* Marre *et al.* [16] used a spanning set of entities to generate test suites in order to estimate and to reduce the cost of testing based on the observation that one test case generally covers more than one entity. As a result, if we can identify “a subset of entities with the property that any set of tests covering this subset covers every entity in the program,” then we can reduce the cost of testing. They presented a method for finding a minimum set of entities of full coverage and an automated method for finding the corresponding spanning set.

Ling *et al.* [14] proposed a decision-tree learning algorithm to build more effective decision trees in order to minimize the sum of the misclassification cost and the test cost. Their algorithm is based on cost-sensitive learning methods such as a Markov Decision Process. They also explained the problems of other approaches and claimed that their algorithm is superior to the others.

Black *et al.* [2] used bi-criteria, two criteria of the problem of interest, based on the interest of the test suite minimization and the ability of the test suite to detect errors. A binary integer linear programming (ILP) is used to combine these two objects into a single-objective mathematical model. By using a binary ILP representation of the test suite minimization problem, the authors compute optimal minimized test sets with maximized error detection rate.

*Adaptive Testing:* Adaptive Random Testing is an active research topic because of its effectiveness under some well distributed input domains [6] The actual results

depend on the “distances” between different test values. However, such distances have only been defined for integers and other elementary values. Ciupa *et al.* [7] extended this idea by introducing an “object distance” to test object-oriented programs. A Distance-Based Adaptive Random Testing (D-ART) method was also proposed based on object distance.

Mayer [17] explained the weakness of D-ART on higher-dimensional input domains and proposed ID-D-ART (Increasing Domain D-ART) which selects the test cases from a sub-domain of an input domain instead of from the entire input domain. The sub-domain increases monotonically until it reaches the entire input domain. Due to the similarity of D-ART and RRT, the applicability of ID-D-ART into the RRT (Restriction-based Random Testing) is also discussed.

Studies such as [13] reported methods for component-based testing in an adaptive manner with the objective to reduce test costs. The proposed method used three key concepts: software cybernetics; a recursive least squares method; and controlled Markov chains. These concepts are used to create a closed-loop feedback which can be used to adjust parameter estimation. These parameters are then fed into the controller which adjusts the necessary testing strategy introduced as estimating software parameters such as failure detection rate. All the testing history is stored in an on-line database.

*Component-based Testing:* Damm *et al.* [8] proposed a framework for automated component testing. This approach is based on Test-Driven Development (TDD) which creates the test cases before developing software components. The proposed framework extends the traditional TDD (which is for each class and method) to the component level, which needs to test for each component interface. As a result, defects can be detected earlier in the development cycle to reduce the overall cost of testing and debugging.

One difficulty of testing software components among others is testing them under numerous hardware, operating systems, and third-party COTS components. Grundy *et al.* [12] proposed an approach using a “validation agent” and a “component aspect” to resolve this problem. They are used at the deployment time to validate the components. The validation agent tests functional and non-functional aspects of software components in an actual deployment situation, whereas the component aspect cross-cuts the aspects of the components to increase its usability.

Component adequate testing is very challenging because each component has a different technology background and uses different component usage patterns as well as reuse contexts. Gao *et al.* [10] focused on component test coverage to solve these issues. They proposed two test models, CFAGs (a component black box test model) and D-CFAGs (a dynamic black box test model), to represent a component’s APL-based function accessing pattern in both static and dynamic views. These test models are used for component validation with respect to related test coverage criteria.

Currently, the assurance of components is certified by a third-party certification. However, most of the certification is only done by limited non-functional testing. Ma *et al.* [15] pointed out two problems in the current approach: lack of information on “future execution context and various users’ usage patterns” and “no access to the source code of components.” They proposed a framework for testing components which include the following three steps: providing guidelines and supporting tools; generating test packages; and executing tests and generating a report. They claimed that their framework gives more power to the third-party certification method.

## 6. Conclusions and Future Work

A new approach that selects a reduced set of test cases for the accurate performance evaluation of components has been presented in this work. The approach is based on the use of polynomial curve fitting techniques. The approach begins by using a small set of initial test cases. Using an iterative approach, a new test case is found and added to the test case set. The new test case may be selected either randomly or using an adaptive technique. The test cases are executed; the performance evaluation results are used to compute a new curve. When the RMSE between the previous and the current curves are below a threshold, then the reduced test case set has been found.

The approach has been experimentally validated using a set of components that implement well-known image compression algorithms: Arithmetic Encoding, BWT, and Huffman. Two non-functional attributes are evaluated for these components. The first is compression ratio; the second is compression time. Using our approach, as few as eight test cases are needed and selected in this experiment to capture the dominant behavior of the two non-functional behaviors. The selected test cases have a root mean square error of 0.0254 in comparison to the actual behavior of the component evaluated with a comprehensive set of 190 test cases. These results indicate the accuracy of the approach is excellent and offers a significant reduction in the number of test cases.

The performance of the new approach has also been experimentally evaluated, comparing the random selection of an additional test case with the adaptive approach. Our results indicate that non-linear behavior, e.g., compression ratio, has better performance with the adaptive approach; linear behavior, e.g., compression time, has better performance with a random approach.

There are several interesting directions for future work. The first is to apply the approach to additional sets of components and evaluate different non-functional attributes, to improve the validation of the approach. The second is to investigate the impact of the values of (1) the degree of the polynomial used in the curve fitting calculation and (2) the error threshold. When the impact of these values are thoroughly quantified, then the values for non-functional attributes could be optimized.

## References

1. *Introduction to Algorithms* (MIT Press and McGraw-Hill, 2001), Sec. 16.3, pp. 385–392.
2. J. Black, E. Melachrinoudis and D. Kaeli, Bi-criteria models for all-uses test suite reduction, *Proceedings of the 26th International Conference on Software Engineering*, pp. 106–115.
3. M. Burrows and D. J. Wheeler, A block-sorting lossless data compression algorithm, Technical Report 124, Digital Systems Research Center, Palo Alto, California, May 1994.
4. J. W. Cangussu, K. Cooper and E. Wong, Reducing the number of test cases for performance evaluation of components, *Proceedings of the 19th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, Boston, USA, 9–11 July 2007.
5. D. Carney and F. Long, What do you mean by cots? Finally, a useful answer, *IEEE Software* **17**(2) 2000 83–86.
6. T. Y. Chen, H. Leung and I. K. Mak, Adaptive random testing, in *Lecture Notes in Computer Science*, 3321, 2004, pp. 320–329.
7. I. Ciupa, A. Leitner, M. Oriol and B. Meyer, Object distance and its application to adaptive random testing of object-oriented programs, *Proceedings of the 1st International Workshop on Random Testing*, July 2006.
8. L. Damm and L. Lundberg, Results from introducing component-level test automation and test-driven development, *Journal of Systems and Software*, 2006.
9. W. Gander, J. H. Masaryk and J. Hrebicek, *Solving Problems in Scientific Computing Using Maple and MATLAB* (Springer-Verlag, 1997).
10. J. Gao, R. Espinoza and J. He, Testing coverage analysis for software component validation, *Proceedings of the 29th Annual International Computer Software and Applications Conference*, July 2005.
11. W. Gautschi, *Numerical Analysis: An Introduction* (Birkhauser, Boston, Cambridge, 1997).
12. J. Grundy, G. Ding and J. Hosking, Deployed software component testing using dynamic validation agents, *Journal of Systems and Software*, 2005.
13. H. Hu, W. E. Wong, C. Jiang and K. Cai, A case study of the recursive least squares estimation approach to adaptive testing for software components, *Proceedings of the 5th International Conference on Quality Software*, September 2005.
14. C. X. Ling, Q. Yang, J. Wang and S. Zhang, Decision trees with minimal costs, *Proceedings of the 21st International Conference on Machine Learning*, July 2004.
15. Y. Ma, S. Oh, D. Bae and Y. Kwo, Framework for third party testing of component software, *Proceedings of the 8th Asia-Pacific Software Engineering Conference*, December 2001.
16. M. Marre and A. Bertolino, Reducing and estimating the cost of test coverage criteria, *Proceedings of the 18th International Conference on Software Engineering*, May 1996.
17. J. Mayer, Towards effective adaptive random testing for higher-dimensional input domains, *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, July 2006.
18. G. A. F. Seber and C. J. Wild, *Nonlinear Regression* (John Wiley & Sons, 2006).
19. G. A. F. Seber and A. J. Lee, *Linear Regression Analysis*, 2nd ed. (Wiley-Interscience, 2003).
20. I. H. Witten, R. Neal and J. G. Cleary, Arithmetic coding for data compression, *Communications of the Association for Computing Machinery*, 1987.