WL–TR–94–1102

MACHINE LEARNING: A COMPARATIVE STUDY OF PATTERN THEORY AND C4.5

MR JEFFREY ALAN GOLDMAN

SYSTEMS CONCEPTS SECTION
MISSION AVIONICS DIVISION

JUNE 1994
FINAL REPORT FOR 12/01/93–06/01/94

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT PATTERSON AFB OH 45433–7409
NOTICE

When Government drawings, specifications or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility nor any obligation whatsoever. The fact that the government may have formulated, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise in any manner construed, as licensing the holder or any other person or corporation, or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

JEFFREY ALAN GOLDMAN
Computer Scientist
Technology Section

DOUGLAS S. HAGER, Chief
Technology Section
Target Recognition Technology Branch

WILLIAM E. MOORE, Acting Chief
Mission Avionics Division
Avionics Directorate

If your address has changed, if you wish to be removed from our mailing list or if the addressee is no longer employed by your organization, please notify WL/AART, Bldg 22, 2690 C Street STE 1, Wright-Patterson AFB, OH 45433-7408 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.
The Machine Learning field has identified several different inductive bias classes with Occam's Razor being held as an accepted paradigm. C4.5, an extension of ID3, is one of the leaders in this class of learning systems with which other systems measure their ability. A completely different approach, yet still a method in the class of Occam biased learning mechanisms, is Pattern Theory. This approach seeks to recognize patterns in a robust manner using function decomposition. FLASH, the embodiment of Pattern Theory is itself, an inductive learning system. In this study, we hope to show that the Pattern Theoretic approach is not only as good as the classic decision tree methods, but also it exhibits strong promise to be a robust technique to identifying patterns. The comparisons will be made by constructing learning curves for each system.
Contents

1 Background Into The Pattern Theory Approach
2 The C4.5 System
3 Description of Benchmark Set
   3.1 RANDOM
   3.2 RANDOM MINORITY ELEMENTS
   3.3 BOOLEAN EXPRESSION
   3.4 VARIATION ON THE MONK PROBLEMS
   3.5 STRING FUNCTIONS
   3.6 IMAGES
   3.7 SYMMETRIC FUNCTIONS
   3.8 NUMERICAL FUNCTIONS
4 Experimental Design
5 Experimental Results
6 Analysis
7 Conclusion and Summary
8 Future Work
A Individual Learning Curves of Each Function for C4.5 and FLASH
B Listing of the Decomposition Plan

List of Figures
1 Function on four variables ........................................ 1
2 Decomposed function on four variables .......................... 2
3 Generalisation Comparison ....................................... 15

List of Tables
1 C4.5 trials with default options, varying the number of trees ........................................ 6
2 C4.5 trials with window size 0, varying the number of trees ........................................ 7
3 C4.5 trials with threshold size 0, varying the number of trees ........................................ 8
4 C4.5 trials with window and threshold size 0, varying the number of trees .......................... 9
5 C4.5 trials with varying options .................................. 10
6 C4.5 trials with grouping options ................................ 11
7 C4.5's best options with FLASH's best .......................... 14
1 Background Into The Pattern Theory Approach

The Pattern Theory paradigm focuses on two central ideas shown in this section. The first is functions that the investigator wishes to learn, have low decomposed function cardinality. The second is functions with low decomposed function cardinality are learnable with a relatively small number of samples. In this section, we will present some background on function decomposition and how Pattern Theory uses this as a robust way to find patterns.

Decomposing a function involves breaking it up into smaller subfunctions. These smaller functions are further broken down until all subfunctions will no longer decompose. For a given function, the number of ways to choose two sets of variables (the partition space) is exponential. The decomposition space is even larger, since there are several ways the subfunctions can be combined and there are several levels of subfunctions possible. The complexity measure that we use to determine the relative predictive power of different function decompositions is called Decomposed Function Cardinality (DFC).

DFC is calculated by adding the cardinalities of each of the subfunctions in the decomposition. The cardinality of an n-variable binary function is $2^n$. We illustrate the measure in the above figures. In Figure 1, we have a function on four variables with cardinality $2^4 = 16$. In Figure 2, we show the same function after it has been decomposed. The DFC of this representation for the original function is $2^2 + 2^2 + 2^2 = 12$. The DFC measures the relative complexity of a function. When we search through the possible decompositions for a function, we choose one with the smallest DFC. This decomposition is our learned concept.

The decomposed representation of the function is one that exhibits more information than the alternative. For example, Figure 1 is essentially a lookup table of inputs and outputs. Figure 2, on the other hand, is a function that is not simply a table. The decomposition, for example, could be two simple functions combined together.

Throughout the paper when we refer to a minimal function decomposition, we use “minimal” to mean a decomposition such that the DFC is the smallest possible for the entire set of decompositions. It is noted that a given minimal decomposition is not unique. For a more rigorous explanation of the inner workings of function decomposition or function extrapolation, the reader is referred to [1], [2] and [8].

An important point is that a function with a low DFC has been experimentally and theoretically determined to be learnable with a small number of samples [8]. Also, functions we are interested in learning, (i.e., functions that are highly “patterned,”) have a low DFC. The Function Learning And Synthesis Hot-Bed (FLASH) was developed to explore function decomposition, and pattern finding. This paper will show that the FLASH program exhibits promising results for finding patterns robustly.

---

1 System Concepts, Wright Laboratory, WL/AART-2 2690 C Street STE 1, Wright-Patterson AFB, Ohio 45433-7408 Email: goldmanj@ase.wpxfs.of.mil
2 The C4.5 System

C4.5 is a machine learning software package. A detailed study of it is given in [7]. The intention of this section is to familiarise the reader with general information about how C4.5 learns a concept. It is important to note that C4.5 is equipped to handle noisy data, conflicting data, continuous variables, and other features which are not our primary concern in this discussion. Although pattern theory is interested in these issues, we are testing performances given binary variables and 100% truthful data.

C4.5 is a decision-tree and rule-based system. This makes it a shallow reasoner. In other words, a deep understanding of the world is not required. The advantages of a shallow reasoner are in the separation of knowledge and control, there is a natural mapping to rules, the rules are modular, and it is easy to provide an explanation. The disadvantages are the brittleness associated with an implicit domain model, it lacks common sense, it lacks robustness, there are problems with formal verification, and often they have limited learning capabilities. A rule-based system works best with diagnosis, configuration and control, and process control. Moreover, rule-based systems are excellent for any system that has independent states, simple control flow (limited branching), and the ability to state knowledge needed without stating how it was obtained.

The C4.5 system has many different options that can be altered by the user for a given learning environment. The default options are as follows. First, given a training set, C4.5 builds a decision tree using the gain ratio criterion. In short, C4.5 chooses a test for the tree if it splits the training data into two unbalanced groups (i.e., Only Positive, Only Negative, Largest Positive). The measure is also normalised. In essence, the gain metric is a measure of Entropy. Second, C4.5 has a threshold default of 2 for a given test in a tree. The test must have at least 2 outcomes with a minimum number of cases. To be more precise, the sum of the weights of the cases for at least two subsets must attain a minimum of 2. We would increase this value if we had noisy data.

Other flexibility built into C4.5 includes changing the amount of pruning of the decision tree (for more generalisation and better predictability with noisy data), allowing C4.5 to choose among n best trees, windowing, debugging, use of continuous variables, using the older unnormalised gain, and various options for the rule induction program. For our purposes, we will not be concerned with pruning since we are interested in C4.5's best classification of the training data. We will, however, test different weight minimums for the gain metric, vary the number of trees, test grouping, and change minimums and maximums for windowing sizes.

Windowing in C4.5 is a feature that is used when creating the initial tree in the test cases. The procedure is to select a random number of training cases and build a tree. This tree is then used to classify the remaining training cases. Any misclassifications are used in a new refinement of the original tree. The cycle is repeated until a tree is built that correctly classifies all of the training data. C4.5 allows you to alter the number of cases to be included in the initial window. It also lets you specify a maximum number of cases that can be added to the window at each iteration. The grouping option for C4.5 allows the method to group discrete attributes by value. Quinlan describes this procedure in detail when we have discrete variables with many values. The purpose of grouping is to prevent forced binary splits. It uses an iterative merging technique on the training elements. We were uncertain at the time of testing, if this grouping would have any relevance to our binary variable domain.
Therefore, to be thorough, it was better to try the method than to ignore it.

It is useful at this point to discuss the option that allows C4.5 to build several trees retaining the best. The reason C4.5 doesn't produce an optimal tree (optimal in the sense that it is the smallest decision tree possible, consistent with the training set) every time is because this problem is NP-complete [5]. Thus, the gain metric is only a heuristic to build a near optimal tree in polynomial time.

Some shortcomings of C4.5 are mentioned in [7]. One is that C4.5 cannot correctly classify cases in which there are non-rectangular regions. For example, in dealing with continuous variables on a two-dimensional plane, the line \( y = x, y > 0 \) does not lend itself to building rectangular regions. Instead, the triangular regions are approximated. Problems arising related to this are poorly delineated regions and fragmented regions. The author attributes fragmented regions to not having enough data to correctly classify.

3 Description of Benchmark Set

Our benchmark set of functions will be used to compare the learning ability of C4.5 and Pattern Theory. This set of functions, although not exhaustive, is designed to include many types of relationships that we might be interested in. The overall goal of testing on several different functions is to compare robust learning ability in this restrictive domain of binary variables. However, it is important to point out that although Pattern Theory is not yet equipped to handle continuous variables, the underlying theory generalises to discrete and continuous variables. The reader is invited to a formal proof and further reading in [8]. The point is key since the binary domain is so restrictive.

The benchmark set includes some 4 dozen functions. The categories break down into: Boolean Expressions, String Functions, Images, Symmetric Functions, Numerical Functions, and Random Functions. All of the functions are of the form \( F : [0, 1]^n \rightarrow [0, 1] \). In other words, there are eight binary variable inputs and one binary variable output. A detailed description of each function is given here.

3.1 RANDOM

There are 3 functions that were randomly generated from FLASH with seeds 1,2, and 3. They are labeled: rnd1, rnd2, and rnd3.

3.2 RANDOM MINORITY ELEMENTS

There are 5 functions generated which have a fixed number of minority elements placed at random. The seed for each was 1. They are labeled: rnd_m1, rnd_m5, rnd_m10, rnd_m25, rnd_m50.

3.3 BOOLEAN EXPRESSION

These 10 KDD functions were designed to represent concepts in a database. KDD stands for Knowledge Discovery in Databases. They were first used in [3] and later in [4].

\[
\begin{align*}
KDD1 &= (z_1 z_3) + \overline{z_2} \\
KDD2 &= (z_1 \overline{z_2} z_3) (z_4 + \overline{z}_6) \\
KDD3 &= (z_1 + \overline{z}_2) + (z_1 z_4 z_6) \\
KDD4 &= \overline{z}_4 \\
KDD5 &= (z_1 z_2 \overline{z}_4) + (z_3 \overline{z}_2 z_7 z_8) + (z_1 z_2 z_5 z_6) + (\overline{z}_3 \overline{z}_5) \\
KDD6 &= z_2 + z_4 + z_6 + z_8 \\
KDD7 &= (z_1 z_2) + (z_3 z_4) + (z_5 z_6) + (z_7 z_8) \\
KDD8 &= (z_1 \overline{z}_2) \text{ XOR } (z_1 z_5) \\
KDD9 &= (z_2 \text{ XOR } z_4) (z_1 \text{ XOR } (z_5 z_7)) \\
KDD10 &= (z_1 \Rightarrow z_4) \text{ XOR } (\overline{z}_7 \overline{z}_5(z_2 + z_3))
\end{align*}
\]

multiplexer, mux6, used in Kosa, this is a 2-address bit, 4-data bit multiplexer with two vacuous variables \((z_0\text{ and } z_1)\) to make 8 inputs. Generated mux6 with FLASH and then edited to make mux8. 3/29/94.

"Deep functions" generated by Mike Noviskey: 04.26.94 and or_and_chain8, (removed or_and_chain8 because or_and_chain8(x) = not(and_or_chain8(not(x))), as in DeMorgan's Theorem)
3.4 VARIATION ON THE MONK PROBLEMS

These are 8 binary variable approximation to the Monk's problems of [9].

\( z_1 \): head shape (rnd, octagonal)
\( z_2 \): body shape (rnd, octagonal)
\( z_3 \): smiling (yes, no)
\( z_4, z_5 \): holding (sword, balloon, flag, M16)
\( z_6, z_7 \): jacket color (red, yellow, green, blue)
\( z_8 \): has tie (yes, no)

monkish1: head shape equals body shape or jacket is red.
monkish2: exactly 2 of 6 attributes have 1st value.
monkish3: (jacket green & has sword) or (jacket not blue and body not oct.) generated with FLASH, 4/6/94.

3.5 STRING FUNCTIONS

These functions are operators on 8-bit binary strings. palindrome acceptor; pal, from FLASH 2/18/94.
palindrome output; pal.output, from Mike Novisky, and PVWave, randomly generated 128 bits then mirror imaged them to create the outputs of an 8 variable function. 3/25/94
doubly palindromed output; pal.dbl.output, from Mike as above except he generated 64 bits and flipped them twice. 3/25/94

2 interval acceptors from FLASH 2/18/94;
interval1 accepts strings with 3 or fewer intervals
interval2 accepts strings with 4 or fewer intervals
2 sub-string detectors from FLASH 2/18/94;
substr1 accepts strings with the sub-string “101”
substr2 accepts strings with the sub-string “1100”

3.6 IMAGES

These functions are various bit maps. chXfY means character X from font Y of the Borland font set. All were generated with the Pascal program charfn.exe of 2/28/94.

ch8f0 - kind of a flat plus sign
ch15f0 - an Aztec looking design
ch22f0 - horizontal bar
ch30f0 - solid isosoles triangle
ch47f0 - slash
ch176f0 - every other column of a checker board
ch177f0 - checker board
ch74f1 - triplex J
ch83f2 - small S (thin strokes)
ch70f3 - sans serif F
ch52f4 - Gothic 4

3.7 SYMMETRIC FUNCTIONS

These functions are symmetric, meaning re-arranging the order of the inputs does not affect the output.
parity, from FLASH 2/22/94. contains 4.ones, \((f(x)=1 \text{ if and only if } \text{str} \ x \ \text{has} \ k \ \text{ones})\), from FLASH 3/2/94.
majority.gate, \(f(x)=1\) if and only if \(x\) has more 1’s than 0’s, from FLASH 3/2/94.
3.8 NUMERICAL FUNCTIONS

These functions are various arithmetic operators.

addition; add0, add2, add4 - outputs bits of a 4 bit adder, 0 is the most significant bit, generated with FLASH 2/22/94.
greater-than: \[ f(x_1, x_2) = 1 \text{ if and only if } x_1 > x_2, \] generated with FLASH 3/2/94.
subtraction: subtraction1, subtraction3 - output bits 1 and 3 of the absolute value of a 4 bit difference. 0 is most significant bit, generated with FLASH 3/2/94.
modulus2, output bit 2 of 4-bit modulus 0 is the most significant bit, generated with FLASH 2/22/94.
remainder2, output bit 2 of 4-bit remainder 0 is the most significant bit, generated with FLASH 2/22/94.

4 Experimental Design

The overall design of our experiment is as follows. First, several options were tested on all the benchmark functions in order to determine what parameters yielded the best performance for C4.5. Next, the resulting learning curves were compared with Pattern Theory.

The tests on the individual functions were as follows. First, each method was given a random set of data to train on ranging from 25 to 250 out of a total of 256 possible cases. Once the method was trained, the entire 256 cases were tested and the number of differences were recorded as errors. This procedure was repeated 10 times for a given sample training size in intervals of 25 yielding a maximum, minimum, and average number of errors for each. Thus, the total number of runs for each function was 100 of varying sample size. None of the learning was incremental. All of the runs were independent.

Our first task was to find the best options to maximize C4.5's performance over the entire training set. The options that were varied on C4.5 include the weight (threshold value for branching), initial windowing size, maximum window size, grouping, and the number of trees grown. The results are displayed in Tables 1-6. The rows are for each function tested. The columns are a description of the options given. The value in the table is the average number of errors for a given run, for a given function over the entire sampling of 25 to 250 samples (the average of all 100 points). The value at the bottom is the average number of errors for a given set of options over the entire set of functions. The smaller the number here, the better the overall performance.

The data is divided into six separate tables. Table 1 shows the relative performances of C4.5 with all of the default options, varying the number of trees. The first column is one tree (the default), the second column is 10 trees, and the third column is 100 trees. Table 2 shows the relative performance of C4.5 with the default options except that the windowing size (-w) is set to 0. Again, the three columns vary the number of trees from 1 to 10 to 100. Table 3 has all of the default options except the threshold parameter (-m) is set to 0. Once again, the number of trees are 1, 10, and 100 respectively. Table 4, like the others, has all the default options except the threshold is set to 0 and the window size is set to 0.

Table 5 is slightly different. The first column tests all of the default options with the threshold set to -1. This was compared with the identical run where the threshold was set to 0 (column 1 of Table 3) to ensure that 0 was indeed the lowest possible setting for the threshold parameter. The second column has the default options except the threshold is set to 0 and the maximum number of allowable cases is 256 (-i 256). This means essentially that since this is only an eight variable function, we have no imposed maximum number of cases. Column three is the default options except the threshold is set to 0 and the number of trees built is 10.

Finally, Table 6 shows some final experiments using grouping (-s) in addition to our best options given so far. Column 1 is the default options except the threshold is 0 and the number of trees built is ten. Column 2 is the same except the threshold value is 1.

If we examine all the tables in detail, we can conclude that having the threshold value set to 0 (smallest possible) or 1, there is a significant decrease in the number of errors when compared with the default. There does not appear to be any significant difference between a threshold of 0 and 1. As far as changing window size parameters, there is no significant change. In fact, once we use 10 or more trees, the initial window size parameter does not make any difference. The grouping option does not appear to give any advantage either for this data set. As far as the best number of trees, clearly more is better. However, there doesn't appear to be
<table>
<thead>
<tr>
<th>Function Name</th>
<th>C4.5 Default</th>
<th>C4.5 10 trees</th>
<th>C4.5 100 trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>add0</td>
<td>22.56</td>
<td>22.62</td>
<td>23.12</td>
</tr>
<tr>
<td>add2</td>
<td>64.12</td>
<td>33.5</td>
<td>31.76</td>
</tr>
<tr>
<td>add4</td>
<td>39.72</td>
<td>3.38</td>
<td>3.38</td>
</tr>
<tr>
<td>ch15f0</td>
<td>47.96</td>
<td>37.92</td>
<td>35.62</td>
</tr>
<tr>
<td>ch17f0</td>
<td>20.54</td>
<td>6.4</td>
<td>5.44</td>
</tr>
<tr>
<td>ch177f0</td>
<td>39.14</td>
<td>2.08</td>
<td>2.08</td>
</tr>
<tr>
<td>ch22f0</td>
<td>29.96</td>
<td>15.18</td>
<td>14.6</td>
</tr>
<tr>
<td>ch30f0</td>
<td>18.18</td>
<td>16.06</td>
<td>15.72</td>
</tr>
<tr>
<td>ch47f0</td>
<td>33.08</td>
<td>26.5</td>
<td>26.2</td>
</tr>
<tr>
<td>ch52f4</td>
<td>30.76</td>
<td>28.78</td>
<td>27.96</td>
</tr>
<tr>
<td>ch70f3</td>
<td>15.5</td>
<td>15.08</td>
<td>14.72</td>
</tr>
<tr>
<td>ch74f1</td>
<td>20.9</td>
<td>20.12</td>
<td>19.66</td>
</tr>
<tr>
<td>ch83f2</td>
<td>33.42</td>
<td>32.24</td>
<td>32.42</td>
</tr>
<tr>
<td>ch88f0</td>
<td>20.84</td>
<td>16.84</td>
<td>16.0</td>
</tr>
<tr>
<td>contains_d_ones</td>
<td>80.18</td>
<td>80.6</td>
<td>80.84</td>
</tr>
<tr>
<td>greater_than</td>
<td>21.36</td>
<td>20.96</td>
<td>21.04</td>
</tr>
<tr>
<td>interval1</td>
<td>44.94</td>
<td>44.14</td>
<td>44.58</td>
</tr>
<tr>
<td>interval2</td>
<td>62.82</td>
<td>63</td>
<td>62.14</td>
</tr>
<tr>
<td>kdd1</td>
<td>0.64</td>
<td>0.64</td>
<td>0.64</td>
</tr>
<tr>
<td>kdd10</td>
<td>25.84</td>
<td>23.8</td>
<td>23.52</td>
</tr>
<tr>
<td>kdd2</td>
<td>3.76</td>
<td>3.84</td>
<td>3.6</td>
</tr>
<tr>
<td>kdd3</td>
<td>2.56</td>
<td>1.76</td>
<td>1.76</td>
</tr>
<tr>
<td>kdd4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>kdd5</td>
<td>15.02</td>
<td>14.24</td>
<td>14.96</td>
</tr>
<tr>
<td>kdd6</td>
<td>3.84</td>
<td>3.96</td>
<td>3.9</td>
</tr>
<tr>
<td>kdd7</td>
<td>26.94</td>
<td>28.3</td>
<td>28.46</td>
</tr>
<tr>
<td>kdd8</td>
<td>16.32</td>
<td>6.04</td>
<td>3.2</td>
</tr>
<tr>
<td>kdd9</td>
<td>30.56</td>
<td>16.94</td>
<td>16.2</td>
</tr>
<tr>
<td>majority_gate</td>
<td>48.68</td>
<td>49.16</td>
<td>49.56</td>
</tr>
<tr>
<td>modulus2</td>
<td>16.72</td>
<td>17.42</td>
<td>17.16</td>
</tr>
<tr>
<td>mux8</td>
<td>23.28</td>
<td>17.8</td>
<td>14.52</td>
</tr>
<tr>
<td>pal</td>
<td>18.8</td>
<td>19.16</td>
<td>19.16</td>
</tr>
<tr>
<td>pal_dbl_output</td>
<td>73.74</td>
<td>68.7</td>
<td>65.14</td>
</tr>
<tr>
<td>pal_output</td>
<td>84.74</td>
<td>83.28</td>
<td>82.76</td>
</tr>
<tr>
<td>parity</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>remainder2</td>
<td>31.95</td>
<td>31.49</td>
<td>31.23</td>
</tr>
<tr>
<td>md_m1</td>
<td>11.02</td>
<td>11.24</td>
<td>11.24</td>
</tr>
<tr>
<td>md_m10</td>
<td>29.34</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>md_m5</td>
<td>5.74</td>
<td>5.52</td>
<td>5.52</td>
</tr>
<tr>
<td>md_m50</td>
<td>54.4</td>
<td>55.14</td>
<td>55.08</td>
</tr>
<tr>
<td>md1</td>
<td>87.84</td>
<td>86.46</td>
<td>85.68</td>
</tr>
<tr>
<td>md2</td>
<td>89.04</td>
<td>86.32</td>
<td>85.44</td>
</tr>
<tr>
<td>md3</td>
<td>85.3</td>
<td>84.34</td>
<td>83.68</td>
</tr>
<tr>
<td>substr1</td>
<td>41.44</td>
<td>37.38</td>
<td>36.22</td>
</tr>
<tr>
<td>substr2</td>
<td>33.3</td>
<td>29</td>
<td>26.68</td>
</tr>
<tr>
<td>substr3</td>
<td>64.14</td>
<td>53.02</td>
<td>52.94</td>
</tr>
<tr>
<td>substr4</td>
<td>39.72</td>
<td>3.38</td>
<td>3.38</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>36.23645683</th>
<th>30.87770833</th>
<th>30.34104167</th>
</tr>
</thead>
<tbody>
<tr>
<td>all Functions</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: C4.5 trials with default options, varying the number of trees
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Window Size = 0</th>
<th>Window Size = 0</th>
<th>Window Size = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>add0</td>
<td>23.2</td>
<td>22.62</td>
<td>23.12</td>
</tr>
<tr>
<td>add2</td>
<td>46.84</td>
<td>33.5</td>
<td>31.76</td>
</tr>
<tr>
<td>add3</td>
<td>7.06</td>
<td>3.38</td>
<td>3.38</td>
</tr>
<tr>
<td>ch150</td>
<td>41.68</td>
<td>37.92</td>
<td>35.62</td>
</tr>
<tr>
<td>ch175f0</td>
<td>8.36</td>
<td>6.4</td>
<td>5.44</td>
</tr>
<tr>
<td>ch177f0</td>
<td>7.2</td>
<td>2.08</td>
<td>2.08</td>
</tr>
<tr>
<td>ch22f0</td>
<td>19.46</td>
<td>15.18</td>
<td>14.6</td>
</tr>
<tr>
<td>ch30f0</td>
<td>17.54</td>
<td>16.06</td>
<td>15.72</td>
</tr>
<tr>
<td>ch37f0</td>
<td>31.52</td>
<td>26.5</td>
<td>26.2</td>
</tr>
<tr>
<td>ch52f4</td>
<td>30.04</td>
<td>28.78</td>
<td>27.96</td>
</tr>
<tr>
<td>ch70f3</td>
<td>15.62</td>
<td>15.08</td>
<td>14.72</td>
</tr>
<tr>
<td>ch74f1</td>
<td>20.7</td>
<td>20.12</td>
<td>19.66</td>
</tr>
<tr>
<td>ch83f2</td>
<td>32.72</td>
<td>32.24</td>
<td>32.42</td>
</tr>
<tr>
<td>ch8f0</td>
<td>18.3</td>
<td>16.84</td>
<td>16</td>
</tr>
<tr>
<td>contains_4_ones</td>
<td>80.94</td>
<td>80.6</td>
<td>80.84</td>
</tr>
<tr>
<td>greater_than</td>
<td>20.42</td>
<td>20.96</td>
<td>21.04</td>
</tr>
<tr>
<td>interval1</td>
<td>44.32</td>
<td>44.14</td>
<td>44.58</td>
</tr>
<tr>
<td>interval2</td>
<td>62.74</td>
<td>63</td>
<td>62.14</td>
</tr>
<tr>
<td>kdd1</td>
<td>0.64</td>
<td>0.64</td>
<td>0.64</td>
</tr>
<tr>
<td>kdd10</td>
<td>24.72</td>
<td>23.8</td>
<td>23.52</td>
</tr>
<tr>
<td>kdd2</td>
<td>4</td>
<td>3.84</td>
<td>3.6</td>
</tr>
<tr>
<td>kdd3</td>
<td>3.04</td>
<td>1.76</td>
<td>1.76</td>
</tr>
<tr>
<td>kdd4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>kdd5</td>
<td>15.92</td>
<td>14.24</td>
<td>13.96</td>
</tr>
<tr>
<td>kdd6</td>
<td>3.36</td>
<td>3.36</td>
<td>3.36</td>
</tr>
<tr>
<td>kdd7</td>
<td>27.2</td>
<td>28.3</td>
<td>28.46</td>
</tr>
<tr>
<td>kdd8</td>
<td>9</td>
<td>6.04</td>
<td>3.2</td>
</tr>
<tr>
<td>kdd9</td>
<td>20.18</td>
<td>16.94</td>
<td>16.2</td>
</tr>
<tr>
<td>majority_gate</td>
<td>48.62</td>
<td>49.16</td>
<td>49.56</td>
</tr>
<tr>
<td>modulus2</td>
<td>16.98</td>
<td>17.42</td>
<td>17.16</td>
</tr>
<tr>
<td>mux8</td>
<td>20.4</td>
<td>17.8</td>
<td>14.52</td>
</tr>
<tr>
<td>pal</td>
<td>19.16</td>
<td>19.16</td>
<td>19.16</td>
</tr>
<tr>
<td>pal_dbl_output</td>
<td>71.1</td>
<td>68.7</td>
<td>65.14</td>
</tr>
<tr>
<td>pal_output</td>
<td>84.5</td>
<td>83.28</td>
<td>82.76</td>
</tr>
<tr>
<td>parity</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>remainder2</td>
<td>32.62</td>
<td>31.49</td>
<td>31.23</td>
</tr>
<tr>
<td>md_m1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>md_m10</td>
<td>11.24</td>
<td>11.24</td>
<td>11.24</td>
</tr>
<tr>
<td>md_m25</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>md_m5</td>
<td>5.52</td>
<td>5.52</td>
<td>5.52</td>
</tr>
<tr>
<td>md_m50</td>
<td>54.92</td>
<td>58.14</td>
<td>56.08</td>
</tr>
<tr>
<td>md1</td>
<td>87.14</td>
<td>86.46</td>
<td>85.68</td>
</tr>
<tr>
<td>md2</td>
<td>88.04</td>
<td>86.32</td>
<td>85.44</td>
</tr>
<tr>
<td>md3</td>
<td>85.28</td>
<td>84.34</td>
<td>83.68</td>
</tr>
<tr>
<td>substr1</td>
<td>39.5</td>
<td>37.38</td>
<td>36.22</td>
</tr>
<tr>
<td>substr2</td>
<td>32.42</td>
<td>29</td>
<td>26.68</td>
</tr>
<tr>
<td>subtract1</td>
<td>58.38</td>
<td>53.02</td>
<td>52.94</td>
</tr>
<tr>
<td>subtract3</td>
<td>7.06</td>
<td>3.38</td>
<td>3.38</td>
</tr>
<tr>
<td>Average over</td>
<td>32.44166667</td>
<td>30.87770833</td>
<td>30.34104167</td>
</tr>
<tr>
<td>all Functions</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: C4.5 trials with window size 0, varying the number of trees
<table>
<thead>
<tr>
<th>Function Name</th>
<th>C4.5 1 tree</th>
<th>C4.5 10 trees</th>
<th>C4.5 100 trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>add0</td>
<td>15.93</td>
<td>16.26</td>
<td>16.58</td>
</tr>
<tr>
<td>add2</td>
<td>44.78</td>
<td>27.4</td>
<td>26.32</td>
</tr>
<tr>
<td>add4</td>
<td>27.29</td>
<td>3.6</td>
<td>3.6</td>
</tr>
<tr>
<td>ch15f0</td>
<td>33.25</td>
<td>27.98</td>
<td>25.88</td>
</tr>
<tr>
<td>ch176f0</td>
<td>13.49</td>
<td>5.54</td>
<td>5.54</td>
</tr>
<tr>
<td>ch177f0</td>
<td>26.42</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ch220</td>
<td>22.87</td>
<td>11.63</td>
<td>9.96</td>
</tr>
<tr>
<td>ch30f0</td>
<td>11.87</td>
<td>11.54</td>
<td>12.18</td>
</tr>
<tr>
<td>ch47f0</td>
<td>25.79</td>
<td>21.52</td>
<td>20.92</td>
</tr>
<tr>
<td>ch52f4</td>
<td>23.55</td>
<td>22.63</td>
<td>22.12</td>
</tr>
<tr>
<td>ch70r3</td>
<td>12.17</td>
<td>12.18</td>
<td>12.06</td>
</tr>
<tr>
<td>ch74f1</td>
<td>16.71</td>
<td>15.79</td>
<td>15.8</td>
</tr>
<tr>
<td>ch83f2</td>
<td>27.77</td>
<td>26.03</td>
<td>25.94</td>
</tr>
<tr>
<td>ch8f0</td>
<td>14.56</td>
<td>11.93</td>
<td>11.6</td>
</tr>
<tr>
<td>contains_4_ones</td>
<td>59.1</td>
<td>58.49</td>
<td>58.68</td>
</tr>
<tr>
<td>greater_than</td>
<td>15.89</td>
<td>16</td>
<td>16.03</td>
</tr>
<tr>
<td>interval1</td>
<td>32.95</td>
<td>34.28</td>
<td>33.92</td>
</tr>
<tr>
<td>interval2</td>
<td>44.03</td>
<td>44.35</td>
<td>44.43</td>
</tr>
<tr>
<td>kdd1</td>
<td>0.32</td>
<td>0.32</td>
<td>0.32</td>
</tr>
<tr>
<td>kdd10</td>
<td>17.73</td>
<td>17.52</td>
<td>17.96</td>
</tr>
<tr>
<td>kdd2</td>
<td>2.24</td>
<td>2.76</td>
<td>2.76</td>
</tr>
<tr>
<td>kdd3</td>
<td>1.6</td>
<td>1.28</td>
<td>1.28</td>
</tr>
<tr>
<td>kdd4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>kdd5</td>
<td>11.2</td>
<td>10.52</td>
<td>10.46</td>
</tr>
<tr>
<td>kdd6</td>
<td>2.48</td>
<td>2.48</td>
<td>2.48</td>
</tr>
<tr>
<td>kdd7</td>
<td>19.08</td>
<td>20.69</td>
<td>21.81</td>
</tr>
<tr>
<td>kdd8</td>
<td>10.99</td>
<td>6.35</td>
<td>6.03</td>
</tr>
<tr>
<td>kdd9</td>
<td>21.64</td>
<td>13.79</td>
<td>13.63</td>
</tr>
<tr>
<td>majority_gate</td>
<td>34.85</td>
<td>36.24</td>
<td>35.54</td>
</tr>
<tr>
<td>modulus2</td>
<td>12.15</td>
<td>12.26</td>
<td>12.29</td>
</tr>
<tr>
<td>mux8</td>
<td>19.29</td>
<td>13.96</td>
<td>11.44</td>
</tr>
<tr>
<td>pal</td>
<td>15.96</td>
<td>16.67</td>
<td>16.67</td>
</tr>
<tr>
<td>pal_dbl_output</td>
<td>52.9</td>
<td>50.77</td>
<td>50.13</td>
</tr>
<tr>
<td>pal_output</td>
<td>58.84</td>
<td>58.98</td>
<td>59.39</td>
</tr>
<tr>
<td>path</td>
<td>87.22</td>
<td>86.58</td>
<td>86.99</td>
</tr>
<tr>
<td>remainder2</td>
<td>25.48</td>
<td>25.49</td>
<td>25.36</td>
</tr>
<tr>
<td>md_m1</td>
<td>2.42</td>
<td>2.38</td>
<td>2.38</td>
</tr>
<tr>
<td>md_m10</td>
<td>11.66</td>
<td>11.62</td>
<td>11.62</td>
</tr>
<tr>
<td>md_m25</td>
<td>26.37</td>
<td>25.59</td>
<td>25.59</td>
</tr>
<tr>
<td>md_m5</td>
<td>7.53</td>
<td>6.99</td>
<td>6.99</td>
</tr>
<tr>
<td>md_m50</td>
<td>42.75</td>
<td>42.72</td>
<td>43.01</td>
</tr>
<tr>
<td>md1</td>
<td>60.45</td>
<td>61.5</td>
<td>61.1</td>
</tr>
<tr>
<td>md2</td>
<td>61.52</td>
<td>62.25</td>
<td>61.84</td>
</tr>
<tr>
<td>md3</td>
<td>60.84</td>
<td>60.48</td>
<td>60.84</td>
</tr>
<tr>
<td>subtract1</td>
<td>31.63</td>
<td>30.3</td>
<td>28.67</td>
</tr>
<tr>
<td>subtract2</td>
<td>24.63</td>
<td>22.08</td>
<td>21.13</td>
</tr>
<tr>
<td>subtract3</td>
<td>48.15</td>
<td>41.42</td>
<td>41.43</td>
</tr>
<tr>
<td>Average over</td>
<td>26.406875</td>
<td>23.22375</td>
<td>23.00625</td>
</tr>
<tr>
<td>all Functions</td>
<td>26.406875</td>
<td>23.22375</td>
<td>23.00625</td>
</tr>
</tbody>
</table>

Table 3: C4.5 trials with threshold size 0, varying the number of trees
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Thresh=Wind.=0</th>
<th>Thresh=Wind.=0</th>
<th>Thresh=Wind.=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>add0</td>
<td>15.92</td>
<td>16.26</td>
<td>16.58</td>
</tr>
<tr>
<td>add2</td>
<td>35.4</td>
<td>27.4</td>
<td>26.32</td>
</tr>
<tr>
<td>add4</td>
<td>10</td>
<td>3.6</td>
<td>3.6</td>
</tr>
<tr>
<td>ch15f0</td>
<td>30.8</td>
<td>27.98</td>
<td>25.88</td>
</tr>
<tr>
<td>ch176f0</td>
<td>6.38</td>
<td>5.54</td>
<td>5.84</td>
</tr>
<tr>
<td>ch177f0</td>
<td>5.24</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ch22f0</td>
<td>15.11</td>
<td>11.63</td>
<td>9.96</td>
</tr>
<tr>
<td>ch30f0</td>
<td>12.04</td>
<td>11.54</td>
<td>12.18</td>
</tr>
<tr>
<td>ch47f0</td>
<td>22.62</td>
<td>21.52</td>
<td>20.92</td>
</tr>
<tr>
<td>ch52f4</td>
<td>23.61</td>
<td>22.63</td>
<td>22.12</td>
</tr>
<tr>
<td>ch70f3</td>
<td>11.93</td>
<td>12.18</td>
<td>12.06</td>
</tr>
<tr>
<td>ch74f1</td>
<td>16.52</td>
<td>15.79</td>
<td>15.8</td>
</tr>
<tr>
<td>ch83f2</td>
<td>25.92</td>
<td>26.03</td>
<td>25.94</td>
</tr>
<tr>
<td>ch84f0</td>
<td>13.05</td>
<td>11.93</td>
<td>11.6</td>
</tr>
<tr>
<td>contains_4_ones</td>
<td>58.54</td>
<td>58.49</td>
<td>58.66</td>
</tr>
<tr>
<td>greater_than</td>
<td>16.95</td>
<td>16</td>
<td>16.03</td>
</tr>
<tr>
<td>interval1</td>
<td>33.53</td>
<td>34.28</td>
<td>33.92</td>
</tr>
<tr>
<td>interval2</td>
<td>45.06</td>
<td>44.35</td>
<td>44.43</td>
</tr>
<tr>
<td>kdd1</td>
<td>0.32</td>
<td>0.32</td>
<td>0.32</td>
</tr>
<tr>
<td>kdd10</td>
<td>16.4</td>
<td>17.52</td>
<td>17.96</td>
</tr>
<tr>
<td>kdd2</td>
<td>2.56</td>
<td>2.76</td>
<td>2.76</td>
</tr>
<tr>
<td>kdd3</td>
<td>1.28</td>
<td>1.28</td>
<td>1.28</td>
</tr>
<tr>
<td>kdd4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>kdd5</td>
<td>11.2</td>
<td>10.52</td>
<td>10.46</td>
</tr>
<tr>
<td>kdd6</td>
<td>2.64</td>
<td>2.48</td>
<td>2.48</td>
</tr>
<tr>
<td>kdd7</td>
<td>19.87</td>
<td>20.69</td>
<td>21.81</td>
</tr>
<tr>
<td>kdd8</td>
<td>7.51</td>
<td>6.35</td>
<td>6.03</td>
</tr>
<tr>
<td>kdd9</td>
<td>16.16</td>
<td>13.79</td>
<td>13.63</td>
</tr>
<tr>
<td>majority_gate</td>
<td>35</td>
<td>36.24</td>
<td>35.54</td>
</tr>
<tr>
<td>modulus2</td>
<td>12.06</td>
<td>12.26</td>
<td>12.29</td>
</tr>
<tr>
<td>mux8</td>
<td>17.37</td>
<td>13.96</td>
<td>11.44</td>
</tr>
<tr>
<td>pal</td>
<td>16.35</td>
<td>16.67</td>
<td>16.67</td>
</tr>
<tr>
<td>pal dbl_output</td>
<td>52.16</td>
<td>50.77</td>
<td>50.13</td>
</tr>
<tr>
<td>pal output</td>
<td>58.1</td>
<td>58.98</td>
<td>59.39</td>
</tr>
<tr>
<td>parity</td>
<td>86.47</td>
<td>86.58</td>
<td>86.99</td>
</tr>
<tr>
<td>remaindr2</td>
<td>26.07</td>
<td>25.49</td>
<td>25.36</td>
</tr>
<tr>
<td>md_m1</td>
<td>2.38</td>
<td>2.38</td>
<td>2.38</td>
</tr>
<tr>
<td>md_m10</td>
<td>11.74</td>
<td>11.62</td>
<td>11.62</td>
</tr>
<tr>
<td>md_m25</td>
<td>25.59</td>
<td>25.59</td>
<td>25.59</td>
</tr>
<tr>
<td>md_m3</td>
<td>6.99</td>
<td>6.99</td>
<td>6.99</td>
</tr>
<tr>
<td>md_m50</td>
<td>42.65</td>
<td>42.72</td>
<td>43.01</td>
</tr>
<tr>
<td>md1</td>
<td>61.16</td>
<td>61.6</td>
<td>61.1</td>
</tr>
<tr>
<td>md2</td>
<td>61.63</td>
<td>62.25</td>
<td>61.84</td>
</tr>
<tr>
<td>md3</td>
<td>61.03</td>
<td>60.48</td>
<td>60.84</td>
</tr>
<tr>
<td>substr1</td>
<td>31.12</td>
<td>30.3</td>
<td>28.67</td>
</tr>
<tr>
<td>substr2</td>
<td>24.6</td>
<td>22.08</td>
<td>21.13</td>
</tr>
<tr>
<td>subtract1</td>
<td>45.09</td>
<td>41.42</td>
<td>41.43</td>
</tr>
<tr>
<td>subtract3</td>
<td>10</td>
<td>3.6</td>
<td>3.6</td>
</tr>
</tbody>
</table>

Table 4: C4.5 trials with window and threshold size 0, varying the number of trees
<table>
<thead>
<tr>
<th>Function Name</th>
<th>C4.5 1 tree</th>
<th>C4.5 10 trees</th>
<th>C4.5 10 trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Name</td>
<td>Threshold=1</td>
<td>Threshold=0</td>
<td>Threshold=1</td>
</tr>
<tr>
<td>add0</td>
<td>15.93</td>
<td>16.13</td>
<td>16.26</td>
</tr>
<tr>
<td>add2</td>
<td>44.78</td>
<td>28.69</td>
<td>27.42</td>
</tr>
<tr>
<td>add4</td>
<td>27.29</td>
<td>3.72</td>
<td>3.6</td>
</tr>
<tr>
<td>ch15f0</td>
<td>33.25</td>
<td>28.27</td>
<td>27.87</td>
</tr>
<tr>
<td>ch17f0</td>
<td>13.49</td>
<td>6.25</td>
<td>5.54</td>
</tr>
<tr>
<td>ch17f10</td>
<td>26.42</td>
<td>5.06</td>
<td>0</td>
</tr>
<tr>
<td>ch22f0</td>
<td>22.87</td>
<td>12.02</td>
<td>11.59</td>
</tr>
<tr>
<td>ch30f0</td>
<td>11.87</td>
<td>11.44</td>
<td>11.58</td>
</tr>
<tr>
<td>ch47f0</td>
<td>25.79</td>
<td>20.76</td>
<td>21.52</td>
</tr>
<tr>
<td>ch52f4</td>
<td>23.55</td>
<td>23.23</td>
<td>22.45</td>
</tr>
<tr>
<td>ch70f3</td>
<td>12.17</td>
<td>11.99</td>
<td>12.19</td>
</tr>
<tr>
<td>ch74f1</td>
<td>16.71</td>
<td>16.09</td>
<td>15.83</td>
</tr>
<tr>
<td>ch83f2</td>
<td>27.77</td>
<td>26.77</td>
<td>26.03</td>
</tr>
<tr>
<td>ch84f</td>
<td>14.55</td>
<td>11.9</td>
<td>11.97</td>
</tr>
<tr>
<td>contains_4_ones</td>
<td>59.1</td>
<td>57.81</td>
<td>58.49</td>
</tr>
<tr>
<td>greater_than</td>
<td>15.89</td>
<td>15.55</td>
<td>16</td>
</tr>
<tr>
<td>interval1</td>
<td>32.95</td>
<td>32.94</td>
<td>34.1</td>
</tr>
<tr>
<td>interval2</td>
<td>44.03</td>
<td>45.33</td>
<td>44.2</td>
</tr>
<tr>
<td>kdd1</td>
<td>0.32</td>
<td>0.32</td>
<td>0.32</td>
</tr>
<tr>
<td>kdd10</td>
<td>17.73</td>
<td>17.29</td>
<td>17.4</td>
</tr>
<tr>
<td>kdd2</td>
<td>2.24</td>
<td>2.76</td>
<td>2.76</td>
</tr>
<tr>
<td>kdd3</td>
<td>1.6</td>
<td>1.28</td>
<td>1.28</td>
</tr>
<tr>
<td>kdd4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>kdd5</td>
<td>11.2</td>
<td>11.39</td>
<td>10.52</td>
</tr>
<tr>
<td>kdd6</td>
<td>2.48</td>
<td>2.28</td>
<td>2.48</td>
</tr>
<tr>
<td>kdd7</td>
<td>19.08</td>
<td>20.13</td>
<td>20.7</td>
</tr>
<tr>
<td>kdd8</td>
<td>10.99</td>
<td>7.18</td>
<td>6.35</td>
</tr>
<tr>
<td>kdd9</td>
<td>21.54</td>
<td>14.24</td>
<td>13.79</td>
</tr>
<tr>
<td>majority_gate</td>
<td>34.85</td>
<td>35.75</td>
<td>36.24</td>
</tr>
<tr>
<td>modulus2</td>
<td>12.15</td>
<td>12.29</td>
<td>12.24</td>
</tr>
<tr>
<td>mux8</td>
<td>19.29</td>
<td>14.42</td>
<td>13.72</td>
</tr>
<tr>
<td>pal</td>
<td>15.96</td>
<td>16.77</td>
<td>16.67</td>
</tr>
<tr>
<td>pal_dbl_output</td>
<td>51.9</td>
<td>50.56</td>
<td>50.84</td>
</tr>
<tr>
<td>pal_output</td>
<td>58.84</td>
<td>50.23</td>
<td>58.92</td>
</tr>
<tr>
<td>parity</td>
<td>87.22</td>
<td>86.63</td>
<td>86.58</td>
</tr>
<tr>
<td>remainder2</td>
<td>25.48</td>
<td>26.1</td>
<td>25.45</td>
</tr>
<tr>
<td>md_m1</td>
<td>2.42</td>
<td>2.14</td>
<td>2.38</td>
</tr>
<tr>
<td>md_m10</td>
<td>11.66</td>
<td>11.39</td>
<td>11.62</td>
</tr>
<tr>
<td>md_m25</td>
<td>26.37</td>
<td>25.87</td>
<td>25.59</td>
</tr>
<tr>
<td>md_m5</td>
<td>7.53</td>
<td>7.32</td>
<td>6.99</td>
</tr>
<tr>
<td>md_m50</td>
<td>42.75</td>
<td>42.16</td>
<td>42.72</td>
</tr>
<tr>
<td>md1</td>
<td>60.45</td>
<td>61.38</td>
<td>61.52</td>
</tr>
<tr>
<td>md2</td>
<td>61.52</td>
<td>61.2</td>
<td>62.22</td>
</tr>
<tr>
<td>md3</td>
<td>60.84</td>
<td>60.94</td>
<td>60.5</td>
</tr>
<tr>
<td>subtract1</td>
<td>31.63</td>
<td>29.14</td>
<td>30.03</td>
</tr>
<tr>
<td>subtract2</td>
<td>24.63</td>
<td>22.58</td>
<td>22.33</td>
</tr>
<tr>
<td>subtract3</td>
<td>48.15</td>
<td>42.05</td>
<td>41.44</td>
</tr>
<tr>
<td>subtract4</td>
<td>27.29</td>
<td>3.72</td>
<td>3.6</td>
</tr>
<tr>
<td>Average over all Functions</td>
<td>26.406875</td>
<td>23.37833333</td>
<td>23.2052083</td>
</tr>
</tbody>
</table>

Table 5: C4.5 trials with varying options
<table>
<thead>
<tr>
<th>Function Name</th>
<th>C4.5 10 trees</th>
<th>Threshold=0/grouping (-s)</th>
<th>Threshold=1/grouping (+)</th>
</tr>
</thead>
<tbody>
<tr>
<td>add0</td>
<td>16.26</td>
<td>16.26</td>
<td></td>
</tr>
<tr>
<td>add2</td>
<td>27.4</td>
<td>27.42</td>
<td></td>
</tr>
<tr>
<td>add4</td>
<td>3.6</td>
<td>3.6</td>
<td></td>
</tr>
<tr>
<td>ch16f0</td>
<td>27.98</td>
<td>27.87</td>
<td></td>
</tr>
<tr>
<td>ch17f0</td>
<td>5.54</td>
<td>5.54</td>
<td></td>
</tr>
<tr>
<td>ch17f1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ch22f0</td>
<td>11.63</td>
<td>11.59</td>
<td></td>
</tr>
<tr>
<td>ch30f0</td>
<td>11.54</td>
<td>11.58</td>
<td></td>
</tr>
<tr>
<td>ch47f0</td>
<td>21.52</td>
<td>21.52</td>
<td></td>
</tr>
<tr>
<td>ch52f4</td>
<td>22.63</td>
<td>22.45</td>
<td></td>
</tr>
<tr>
<td>ch70f3</td>
<td>12.18</td>
<td>12.19</td>
<td></td>
</tr>
<tr>
<td>ch74f1</td>
<td>15.79</td>
<td>15.83</td>
<td></td>
</tr>
<tr>
<td>ch83f2</td>
<td>26.03</td>
<td>26.03</td>
<td></td>
</tr>
<tr>
<td>ch88f0</td>
<td>11.93</td>
<td>11.97</td>
<td></td>
</tr>
<tr>
<td>contains_4_ones</td>
<td>58.49</td>
<td>58.49</td>
<td></td>
</tr>
<tr>
<td>greater_than</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>interval1</td>
<td>34.28</td>
<td>34.1</td>
<td></td>
</tr>
<tr>
<td>interval2</td>
<td>44.35</td>
<td>44.2</td>
<td></td>
</tr>
<tr>
<td>kdd1</td>
<td>0.32</td>
<td>0.32</td>
<td></td>
</tr>
<tr>
<td>kdd10</td>
<td>17.52</td>
<td>17.4</td>
<td></td>
</tr>
<tr>
<td>kdd2</td>
<td>2.76</td>
<td>2.76</td>
<td></td>
</tr>
<tr>
<td>kdd3</td>
<td>1.28</td>
<td>1.28</td>
<td></td>
</tr>
<tr>
<td>kdd4</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>kdd5</td>
<td>10.52</td>
<td>10.52</td>
<td></td>
</tr>
<tr>
<td>kdd6</td>
<td>2.48</td>
<td>2.48</td>
<td></td>
</tr>
<tr>
<td>kdd7</td>
<td>20.69</td>
<td>20.7</td>
<td></td>
</tr>
<tr>
<td>kdd8</td>
<td>6.35</td>
<td>6.35</td>
<td></td>
</tr>
<tr>
<td>kdd9</td>
<td>13.79</td>
<td>13.79</td>
<td></td>
</tr>
<tr>
<td>majority_gate</td>
<td>36.24</td>
<td>36.24</td>
<td></td>
</tr>
<tr>
<td>modulus2</td>
<td>12.26</td>
<td>12.24</td>
<td></td>
</tr>
<tr>
<td>mux8</td>
<td>13.96</td>
<td>13.72</td>
<td></td>
</tr>
<tr>
<td>pal</td>
<td>16.67</td>
<td>16.67</td>
<td></td>
</tr>
<tr>
<td>pal_dblo_output</td>
<td>50.77</td>
<td>50.85</td>
<td></td>
</tr>
<tr>
<td>pal_output</td>
<td>58.98</td>
<td>58.92</td>
<td></td>
</tr>
<tr>
<td>parity</td>
<td>86.58</td>
<td>86.58</td>
<td></td>
</tr>
<tr>
<td>remainder2</td>
<td>25.49</td>
<td>25.46</td>
<td></td>
</tr>
<tr>
<td>md_m1</td>
<td>2.38</td>
<td>2.38</td>
<td></td>
</tr>
<tr>
<td>md_m10</td>
<td>11.62</td>
<td>11.62</td>
<td></td>
</tr>
<tr>
<td>md_m25</td>
<td>25.59</td>
<td>25.59</td>
<td></td>
</tr>
<tr>
<td>md_m5</td>
<td>6.99</td>
<td>6.99</td>
<td></td>
</tr>
<tr>
<td>md_m50</td>
<td>42.72</td>
<td>42.72</td>
<td></td>
</tr>
<tr>
<td>md_1</td>
<td>61.5</td>
<td>61.52</td>
<td></td>
</tr>
<tr>
<td>md2</td>
<td>62.25</td>
<td>62.22</td>
<td></td>
</tr>
<tr>
<td>md3</td>
<td>60.48</td>
<td>60.5</td>
<td></td>
</tr>
<tr>
<td>substr1</td>
<td>30.3</td>
<td>30.03</td>
<td></td>
</tr>
<tr>
<td>substr2</td>
<td>22.08</td>
<td>22.33</td>
<td></td>
</tr>
<tr>
<td>subtract1</td>
<td>41.42</td>
<td>41.44</td>
<td></td>
</tr>
<tr>
<td>subtract3</td>
<td>3.6</td>
<td>3.6</td>
<td></td>
</tr>
<tr>
<td>Average over all Functions</td>
<td>23.22375</td>
<td>23.20520833</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: C4.5 trials with grouping options
any significant benefit going from 10 trees to 100. In fact, we have no reason to believe that there will be any significant difference between 10 trees and 1000. Thus our best set for which we will test against Pattern Theory will be with all the options set at default except the weight (threshold) will be 0 (-m 0) and the number of trees will be 10 (-t 10). We will use -m 0 over -m 1 because this is the lowest setting we can have that corresponds to not having any noise in the data.

5 Experimental Results

Now that we have the best options possible for C4.5 over our benchmark set of functions, we can test it with honesty against Pattern Theory. This section refers to the learning curves for every function tested. The curves themselves are shown in Appendix A. The sets are displayed with C4.5 first on a given function, then Pattern Theory. For a given graph, the Y-axis is the number of errors and the X-axis is the number of training samples. Each graph includes the maximum, minimum, and average error. The experiments stopped if the maximum error reached 0 (i.e., for all 10 runs, there were no errors). Thus, there would not be any data points beyond that particular sample size. The chance line, represented by dashes, is the error expected if we were to randomly guess on the remaining cases. We would expect to get half of them right and half wrong since there are only two outcomes. On the graphs for FLASH (Pattern Theory), there are additional points plotted corresponding to the calculated DFC and the number of "don't cares" for a given sampling. We can also see that functions that are highly patterned have a low DFC while more complicated patterns have a higher DFC. Moreover, the random functions have a very high DFC.

Earlier, we gave some background about how function decomposition works and thus how FLASH works. What has not been described is the actual search procedure that FLASH uses in order to select a partition. First, the same strategy was used for every experiment. Essentially, it is a two-ply look ahead on all possible partitions. The calculated DFC is used to continue selecting partitions until they no longer decompose. The actual strategy itself is given in Appendix B. The name of the decomposition plan is dniOe300.

6 Analysis

If we examine C4.5's performance as a whole, its ability ranges from extremely good to extremely poor. C4.5's performance was excellent for the Boolean Expression functions; it had a respectable performance for PAL, MUX8, MODULUS2, and GREATER_THAN; it has a hard time with the other string functions, and it is poor at learning the other PAL functions. C4.5 is especially poor at PARITY. Its performance on the character functions are mixed. Some it learns very well and others, the performance is fair. A few anomalies were the fact that C4.5 performed very well on ADD4 and ADD0 but poorly on ADD2. It was even more bizarre to see excellent performance on SUBTRACTION3 and very poor performance on SUBTRACTION1.

Comparing C4.5 and FLASH (Pattern Theory/Function Decomposition), C4.5 beats FLASH for only two functions: KDD2 and KDD3. It is equal or slightly better for the two character functions: CH52F4 and CH83F2. For all of the other functions however, FLASH outperforms C4.5. In some cases, the performance margin is substantial. The notable cases are: ADD2, ADD4, CONTAINS_AONES, KDD7, KDD9, KDD10, MAJORITY_GATE, PARITY, SUBTRACTION1, and SUBTRACTION3. Of course, we are not concerned with comparing performance on different random functions. Their purpose is to measure consistency and normal behavior. It would be unusual for any method to be significantly better in some random function than another.

The other functions were not mentioned here because some comparisons might be construed as unobjective. Although different performance is seen in some cases, in general, we see equal performance or FLASH performing better. The above functions were mentioned specifically because of the vast differences between the two programs.

In general what we see in C4.5 is that it is unequipped to handle "XOR" type relations. The inherent problem is its inability to deal with replication in such disjunctive concepts as: (A and B) or (C and D). This is as expected [8]. It would appear that C4.5 is unable to effectively learn functions that have an "XOR" or lend themselves to "XOR." FLASH, on the other hand, has no restrictions in this area. There are still some problems with functions that have deep replication that prevent FLASH from completely learning such a function unless all of the samples are given. However, its performance does not degrade beyond C4.5.
C4.5 holds its own for the Boolean functions that do not involve “XOR.” It also had a respectable performance for some of the character functions. However, the best domain for C4.5 is the class of Boolean Expressions. In the other areas, it does not stand up to FLASH.

In Table 7, we list the average errors for all the functions, similar to the previous section. Here, however, we show C4.5's best with FLASH. One can see from the total average error that FLASH is outperforming C4.5 as a robust pattern finder in this domain of binary variables and noise free data. This table also shows off to the side, how the average error changes after successively removing functions that C4.5 is unable to handle. Once they are all removed, their respective performances are nearly equal.

It is noted to the reader that although the Table 7 provides a nice compact comparison, it is not 100% reliable. There are a few functions in which the average error does not correspond to performance. They are anomalous and are few in number. They are notably KDD2, Subtraction3, and ADD4. For example in KDD2, C4.5's average error is 2.76 versus FLASH's 2.4. However, C4.5 learns the function in 125 samples and FLASH learns the function in 150 samples. On the other hand in Subtraction3, C4.5's average is 3.6 versus FLASH's 0. The two performances appear similar. But, C4.5 learns the function in 150 samples and FLASH learns the function in only 25 samples. For a comprehensive analysis, the reader is again referred to the individual learning curves in Appendix A.

Another attempt is made to summarise all of the data from the graphs in Appendix A shown in Figure 3. Here, we show the number of functions learned versus the number of samples for FLASH and C4.5. From the figure, we can see a clear performance distinction between the two methods.

7 Conclusion and Summary

In conclusion, FLASH (Pattern Theory) was shown to be a more robust pattern finder than C4.5 for our limited domain of binary variables and noise free data. Again, we emphasise the point that Pattern Theory can be extended to discrete and continuous valued variables demonstrating its flexibility. C4.5 held its own in the Boolean Expression domain and some of the images, however, its performance was lacking in comparison to the other domains. Specifically, C4.5 fails to learn concepts with implicit “XOR” representations or functions that have duplication in their subtrees.

Pattern Theory has been demonstrated a robust, effective inductive learning technique comparable to the best. The experimental results show its learning ability relative to chance and C4.5. Furthermore, as displayed by our graphs, there is a correlation between a function that is highly “patterned” and a function that has a low DFC.

8 Future Work

Some future directions in this area are to continue testing more functions like those in our experiments. In fact, a few of the functions tested were added after most of the experiments were performed (the monk problems and the “deep functions”) and were not included in the first 7 tables. New functions are constantly being tested, but we had to wrap up the discussion at some point. However, their graphs were included in the Appendix A for study. In addition, it is planned to increase the number of variables to as many as 30 in the immediate future. We are also looking into adapting the current program to handle discrete and continuous variables. Furthermore, we ultimately plan to incorporate methods of handling noise. Moreover, we are looking for ways to increase the speed by limiting the exploration of the partition search space.

We have a working theoretical result of applying function decomposition to continuous variables and the searching ability is getting better. At this point, the real limitation is the number of variables and noise. Since function decomposition involves an exponential search space, the only hope is using some method to prune the branches of the tree. At the rate the work is going, it is very possible that at the time of this printing, we will be able to handle up to 100 variables with the same accuracy.

Noise, on the other hand, is a more difficult problem. At present, we have no formal theoretical basis for dealing with it. It is, however, a personal interest of the author and the hope is to perform some quality research in this area.
<table>
<thead>
<tr>
<th>Function Name</th>
<th>C4.5 10 trees</th>
<th>Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Threshold = 0</td>
<td></td>
</tr>
<tr>
<td>addO</td>
<td>16.26</td>
<td>10.38</td>
</tr>
<tr>
<td>add2</td>
<td>27.4</td>
<td>5.24</td>
</tr>
<tr>
<td>add3</td>
<td>3.6</td>
<td>0</td>
</tr>
<tr>
<td>ch150</td>
<td>27.98</td>
<td>19.595</td>
</tr>
<tr>
<td>ch176f0</td>
<td>5.54</td>
<td>0.16</td>
</tr>
<tr>
<td>ch177f0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ch22f0</td>
<td>11.63</td>
<td>7</td>
</tr>
<tr>
<td>ch30f0</td>
<td>11.54</td>
<td>10.29</td>
</tr>
<tr>
<td>ch47f0</td>
<td>21.52</td>
<td>16.89</td>
</tr>
<tr>
<td>ch52f4</td>
<td>22.63</td>
<td>27.74</td>
</tr>
<tr>
<td>ch70f3</td>
<td>12.18</td>
<td>12.04</td>
</tr>
<tr>
<td>ch74f1</td>
<td>15.79</td>
<td>15.85</td>
</tr>
<tr>
<td>ch83f2</td>
<td>26.03</td>
<td>27.885</td>
</tr>
<tr>
<td>ch8f0</td>
<td>11.93</td>
<td>11.7</td>
</tr>
<tr>
<td>contains_4_ones</td>
<td>58.49</td>
<td>24.49</td>
</tr>
<tr>
<td>greater_than</td>
<td>16</td>
<td>9.78</td>
</tr>
<tr>
<td>interval1</td>
<td>34.28</td>
<td>33.585</td>
</tr>
<tr>
<td>interval2</td>
<td>44.35</td>
<td>35.94</td>
</tr>
<tr>
<td>kdd1</td>
<td>0.32</td>
<td>0</td>
</tr>
<tr>
<td>kdd10</td>
<td>17.52</td>
<td>8.18</td>
</tr>
<tr>
<td>kdd2</td>
<td>2.76</td>
<td>2.4</td>
</tr>
<tr>
<td>kdd3</td>
<td>1.28</td>
<td>2.72</td>
</tr>
<tr>
<td>kdd4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>kdd5</td>
<td>10.52</td>
<td>11.11</td>
</tr>
<tr>
<td>kdd6</td>
<td>2.48</td>
<td>3.72</td>
</tr>
<tr>
<td>kdd7</td>
<td>20.69</td>
<td>10.53</td>
</tr>
<tr>
<td>kdd8</td>
<td>6.35</td>
<td>0</td>
</tr>
<tr>
<td>kdd9</td>
<td>13.79</td>
<td>6.55</td>
</tr>
<tr>
<td>majority_gate</td>
<td>36.24</td>
<td>18.74</td>
</tr>
<tr>
<td>modulus2</td>
<td>12.26</td>
<td>13.4</td>
</tr>
<tr>
<td>mux8</td>
<td>13.96</td>
<td>13.04</td>
</tr>
<tr>
<td>pal</td>
<td>16.67</td>
<td>9.9</td>
</tr>
<tr>
<td>pal_double_output</td>
<td>50.77</td>
<td>38.94</td>
</tr>
<tr>
<td>pal_output</td>
<td>58.98</td>
<td>58.79</td>
</tr>
<tr>
<td>parity</td>
<td>86.58</td>
<td>10.45</td>
</tr>
<tr>
<td>remainder2</td>
<td>25.49</td>
<td>25.22</td>
</tr>
<tr>
<td>md_m1</td>
<td>2.38</td>
<td>2.31</td>
</tr>
<tr>
<td>md_m10</td>
<td>11.62</td>
<td>13.045</td>
</tr>
<tr>
<td>md_m25</td>
<td>25.59</td>
<td>25.61</td>
</tr>
<tr>
<td>md_m5</td>
<td>6.99</td>
<td>7.815</td>
</tr>
<tr>
<td>md_m50</td>
<td>42.72</td>
<td>42.685</td>
</tr>
<tr>
<td>md1</td>
<td>61.5</td>
<td>59.125</td>
</tr>
<tr>
<td>md2</td>
<td>62.25</td>
<td>60.065</td>
</tr>
<tr>
<td>md3</td>
<td>60.48</td>
<td>59.665</td>
</tr>
<tr>
<td>substr1</td>
<td>30.3</td>
<td>24.105</td>
</tr>
<tr>
<td>substr2</td>
<td>22.08</td>
<td>23</td>
</tr>
<tr>
<td>subtract1</td>
<td>41.42</td>
<td>24.22</td>
</tr>
<tr>
<td>subtract3</td>
<td>3.6</td>
<td>0</td>
</tr>
<tr>
<td>Average over all fns</td>
<td>23.22375</td>
<td>17.58521</td>
</tr>
</tbody>
</table>

Table 7: C4.5's best options with FLASH's best
Figure 3: Generalisation Comparison
References


A. Individual Learning Curves of Each Function for C4.5 and FLASH

This section is a set of graphs described in the report. Every graph has the name of the function being tested at the top, the number of errors as the y-axis, and the number of samples as the x-axis.

The tests on the individual functions were as follows. First, each method was given a random set of data to train on ranging from 25 to 250 out of a total of 256 possible cases. Once the method was trained, the entire 256 cases were tested and the number of differences were recorded as errors. This procedure was repeated 10 times for a given sample training size in intervals of 25 yielding a maximum, minimum, and average number of errors for each. Thus, the total number of runs for each function was 100 of varying sample size. None of the learning was incremental. All of the runs were independent.

The chance line was calculated as follows. For a given sample size, assume we simply mimic the data we are given and then randomly guess the remaining unknown elements. This creates a static learning line which represents learning by chance. For a given function, the graph for C4.5 is displayed first followed by FLASH (function decomposition). The FLASH graphs also have some additional information plotted: the average DFC, and the number of “don’t cares” or unknowns. The DFC is calculated by the function realised for each sample size. Since there are ten trials at each sample size, an average DFC is computed.
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

---

Max Error

Min Error

Avg Error
**ADD2**

![Graph showing error vs samples](image)

**Chance**

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

* Max Error

* Min Error

--- Avg Error

---
ADD2

Error

Samples

--- Chance

* Max error

* Min error

--- Avg error

--- Don't cares

--- Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- Max Error
- Min Error
- Avg Error
ADD4

--- Chance

* Max error

* Min error

--- Avg error

--- Don't cares

--- Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- --- Chance

- Max Error

- Min Error

- Avg Error
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)
CH176F0

Error

<table>
<thead>
<tr>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>150</td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>250</td>
</tr>
</tbody>
</table>

Chance

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

Max Error

Min Error

Avg Error
CH176F0

Error

Samples

--- Chance

* Max error

* Min error

--- Avg error

Don't cares

--- Avg DFC

FLASH dni0e300
CH177FO

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

Chance

Max Error

Min Error

Avg Error
CH177F0

Error

Samples

--- Chance

* Max error

* Min error

--- Avg error

........ Don't cares

--- Avg DFC

FLASH dni0e300
CH22F0

Error vs. Samples

--- Chance

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

* Max Error

* Min Error

--- Avg Error
CH22F0

Error

Samples

DFC

--- Chance

* Max error

* Min error

--- Avg error

--- Don't cares

--- Avg DFC

FLASH dni0e300
CH3OFO

--- Chance

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

* Max Error

* Min Error

--- Avg Error

34
CH47F0

C4.5 Threshold=0 (−m 0), 10 Trees (−t 10)

- - - - Chance

Max Error

Min Error

Avg Error
CH47F0

Error

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- Chance

* Max Error

* Min Error

* Avg Error
CH52F4

![Graph showing various error rates and sample counts](image)

- **Chance**: Dashed line
- **Max error**: Solid line
- **Min error**: Dashed-dotted line
- **Avg error**: Solid line
- **Don't cares**: Dotted line
- **Avg DFC**: Dotted line

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- --- Chance

* Max Error

* Min Error

- --- Avg Error
CH7OF3

Samples

Error

DFS

--- Chance

* Max error

* Min error

***** Avg error

--- Don't cares

--- Avg DFC

FLASH dni0e300

41
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

 Chance

 Max Error

 Min Error

 Avg Error
FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- - - - - Chance

* Max Error

* Min Error

* * Avg Error
CH83F2

Error

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

* Max Error

* Min Error

--- Avg Error
CH8FO

Error

Samples

150
200

100

50

0

0

50

100

150

200

250

--- Chance

* Max error

* Min error

--- Avg error

--- Don't cares

--- Avg DFC

FLASH dni0e300
CONTAINS_4_ONES

- - - - - Chance

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

* Max Error

* Min Error

*--* Avg Error

48
CONTAINS_4_ONES

Error

Samples

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300

49
GREATER_THAN

Error

Samples

--- Chance

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

Max Error

Min Error

Avg Error

50
GREATER THAN

Error

Samples

Chance

Max error

Min error

Avg error

Don't cares

Avg D C

FLASH dni0e300
INTERVAL 1

Error

0 50 100 150 200 250

Samples

--- Chance

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

* Max Error

* Min Error

| Avg Error
INTERVAL 1

Error

Samples

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- - - - - Chance

Max Error

Min Error

Avg Error
INTERVAL2

Error

--- Chance
--- Max error
--- Min error
--- Avg error
--- Don't cares
--- Avg DFC

FLASH dni0e300

55
KDD1

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

Chance

Max Error

Min Error

Avg Error
KDD10

Error

Chance

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

Max Error

Min Error

Avg Error

Samples
KDD10

--- Chance

* Max error

* Min error

----- Avg error

----- Don't cares

----- Δ Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- - - - - Chance

Max Error

Min Error

Avg Error
KDD2

Error

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300

61
KDD3

C4.5 Threshold=0 (\texttt{m 0}), 10 Trees (\texttt{t 10})

\begin{itemize}
  \item \texttt{Max Error}
  \item \texttt{Min Error}
  \item \texttt{Avg Error}
\end{itemize}
KDD3

Samples

Error

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
KDD4

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

Chance

Max Error

Min Error

Avg Error
KDD4

Samples

Error

Don't cares

Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

* Max Error

* Min Error

* Avg Error
KDD5

Error

Samples

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- Chance

Max Error

Min Error

Avg Error
KDD6

Error

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- Chance

Max Error

Min Error

Avg Error
KDD7

---

Samples

0 50 100 150 200

Error

0 50 100 150

DNC

---

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
KDD8

![Graph showing error vs. samples]

- C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)
  - Chance
  - Max Error
  - Min Error
  - Avg Error
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- Chance
- Max Error
- Min Error
- Avg Error
KDD9

Error vs. Samples

- - - - - - Chance

* Max error

* Min error

--- Avg error

------ Don't cares

--- Avg DFC

FLASH dni0e300
MAJORITY_GATE

Error

\begin{center}
\begin{tabular}{c}
\text{Majority Gate}
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{c}
\text{Samples}
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{c}
\text{Chance}
\end{tabular}
\end{center}

\text{C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)}

\begin{center}
\begin{tabular}{c}
\text{Max Error}
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{c}
\text{Min Error}
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{c}
\text{Avg Error}
\end{tabular}
\end{center}

76
MAJORITY_GATE

Samples

Error

DFC

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
MODULUS2

Error vs. Samples

--- Chance

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

* Max Error

* Min Error

* Avg Error

78
MODULUS2

Error

Samples

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
MONKISH1

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

Chance

Max Error

Min Error

Avg Error
MONKISH1

---

* Chance

* Max error

* Min error

* Avg error

---

Don't cares

---

Avg DFC

FLASH dni0e300

81
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- - - - Chance

Max Error

Min Error

Avg Error
MONKISH2

Error

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300

83
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)
MONKISH3

Error vs. Samples

--- Chance

* Max error

* Min error

* Avg error

Don't cares

Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)
Samples | Chance | Max error | Min error | Avg error | Don't cares | Avg DFC
---|---|---|---|---|---|---
0 | | | | | | |
50 | | | | | | |
100 | | | | | | |
150 | | | | | | |
200 | | | | | | |
250 | | | | | | |

FLASH dni0e300
OR_AND_CHAIN8

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- - - - - - Chance

Max Error

Min Error

Avg Error
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

Chance

Max Error

Min Error

Avg Error
Samples

Error

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

--- Chance

Max Error

Min Error

Avg Error
PAL.DBLOUTPUT

--- Chance

* Max error

* Min error

--- Avg error

--- Don't cares

--- Avg DFC

FLASH dniOe300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)
PAL_OUTPUT

--- Chance

* Max error

* Min error

--- Avg error

-.--. Don't cares

--- Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

* Max Error

* Min Error

--- Avg Error

PARITY

Samples

Error
Samples Chance Max error Min error Avg error Don't cares Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- Chance

* Max Error

* Min Error

*** Avg Error
REMAINDE12

--- Chance

* Max error

* Min error

--- Avg error

--- Don't cares

--- Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

Max Error

Min Error

Avg Error
RND2

Samples

Error

--- Chance

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

* Max Error

* Min Error

--- Avg Error

102
RND2

Error vs. Samples

- --- - Chance
- * Max error
- * Min error
- * Avg error
- ---- Don't cares
- ---- Avg DFC

FLASH dni0e300
RND3

\begin{figure}
\centering
\includegraphics[width=\textwidth]{chart.png}
\caption{Error vs. Samples for RND3}
\end{figure}

- Solid line: Chance
- Dashed line: C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)
- Star: Max Error
- Star: Min Error
- Star: Avg Error

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)
FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- Chance
- Max Error
- Min Error
- Avg Error
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- - - - - Chance

* Max Error

* Min Error

---* Avg Error
RND_M10

Error

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

--- Chance

* Max Error

* Min Error

--- Avg Error
RND_M25

--- Chance

* Max error

* Min error

--- Avg error

--- Don't cares

--- Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (−m 0), 10 Trees (−t 10)

Max Error

Min Error

Avg Error
RND_M5

Error

Samples

0 50 100 150 200 250

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300

113
RND_M50

Error

Chance

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

Max Error

Min Error

Avg Error
RND_M50

Error

Samples

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300

115
C4.5 Threshold=0 (−m 0), 10 Trees (−t 10)

* Max Error

* Min Error

** Avg Error
SUBSTR1

Error

Samples

0 50 100 150 200 250

0 50 100 150 200 250

---  Chance

*  Max error

*  Min error

---  Avg error

---  Don't cares

---  Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)
SUBSTR2

---

Error vs. Samples

- - - - - Chance

* Max error

* Min error

* Avg error

- - - - Don't cares

- - - - Avg DFC

FLASH dni0e300

119
SUBTRACTION 1

\[ 150 - 100 = 50, \quad 150 - 50 = 100, \quad 100 - 50 = 50, \quad 100 - 100 = 0 \]

Samples

Chance

C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

Max Error

Min Error

Avg Error

120
SUBTRACTION 1

Samples

Error

150

100

50

0

0

50

100

150

200

250

Error

DFC

150

100

50

0

0

50

100

150

200

250

Chance

Max error

Min error

Avg error

Don't cares

Avg DFC

FLASH dni0e300
C4.5 Threshold=0 (-m 0), 10 Trees (-t 10)

- - - - - Chance

Max Error

Min Error

Avg Error
SUBTRACTION 3

Error

Samples

--- Chance

* Max error

* Min error

--- Avg error

...... Don't cares

--- Avg DFC

FLASH dni0e300
B Listing of the Decomposition Plan

Appendix B shows the actual decomposition plan used in our tests with FLASH. The listing is a printout of the file dni0e300.
Decomp Plan:
Selection Plan:
) = use shared variables
.2 = method
) = first part type
) = stopping condition
Evaluation Plan:
) = no of partition tests
) = measure challenger by
) = measure champ by
) = threshold in n
) = champ_multiplier
) = dp_for_children_is_same
Decomp Plan:
Selection Plan:
) = use shared variables
.2 = method
) = first part type
) = stopping condition
30 = stopping condition parameter
Evaluation Plan:
) = no of partition tests
) = measure challenger by
) = measure champ by
) = threshold in n
) = champ_multiplier
) = measure challenger by
) = measure champ by
) = threshold in n
) = champ_multiplier
) = Random_No generator seed (>0)
) = dp_for_best_part_children_is_same
Decomp Plan:
Selection Plan:
) = use shared variables
12 = method
2 = first part type
1 = stopping condition
1 = stopping condition parameter
Evaluation Plan:
) = no of partition tests
) = measure challenger by
) = measure champ by
) = threshold in n
) = champ_multiplier
) = Random_No generator seed (>0)
) = dp_for_best_part_children_is_same
) = Random_No generator seed (>0)
) = dp_for_best_part_children_is_same

125