AN ALGORITHMIC APPROACH TO THE
DETECTION AND PREVENTION OF
PLAGIARISM

Karl J. Ottenstein
Computer Sciences Department
Purdue University
West Lafayette, Indiana 47907

CSD-TR 200
August 1976
The significant problem of detecting (nearly) identical student homework papers is non-trivial since a grader for a large class cannot remember all previously graded papers while examining the current one. This problem can be reduced by quantifying papers in such a way that equivalent ones are given equal values. Here we discuss one possible quantification which works well when applied to student computer programs.

The desired quantification is a function which maps the "homework space" into some value space. The ideal function, \( f \), would impose a partitioning on the set of papers in the sense that if \( x \) and \( y \) are homework papers and the \( P_i \) are partitions of the homework space with \( x \in P_i \) and \( y \in P_j \) then \( i = j \) iff \( f(x) = f(y) \). If \( f(x) = f(y) \) and \( x \) and \( y \) are unique, then one of \( x \) and \( y \) is a plagiarized version of the other. In other words, when all partitions have but one element, no cheating has occurred. This ideal function is unobtainable for several reasons: it is possible for identical work to be performed independently, the semantic equivalence of two items cannot always be shown deterministically, and there is a subjective area between plagiarism and paraphrasing.

Our task, then, is to find a good approximation to this function. The approximation should at least map all potentially equivalent homework papers into the same partition. It may not guarantee accuracy in that two papers being in the same partition will not imply that they are necessarily plagiarized. If \( P_1, P_2, \ldots, P_n \) are the ideal partitions, our approximation should create \( Q_1, Q_2, \ldots, Q_m \) where each \( Q_i \) is either some \( P_j \) or the union of several \( P_j \)'s. That is, the partitions are merely cruder.

The constant functions satisfy our requirements for an approximation since only one partition will be created; but, they do not simplify our initial problem since all elements must be individually inspected for cheating. A function which maps a homework paper into the integer representing its length in characters will invariably create numerous partitions, but they will not be the desired \( Q_i \): the replacement of one token by a
A synonym of a different length will place plagiarized assignments in separate partitions. A length function based on the number of tokens would eliminate this problem, but will still group together totally unrelated assignments simply because they have the same length. A function which takes into account some measure of the information content of a homework paper should give us more accurate partitions.

Any meaningful language can have its symbols classified into three sets:

- operators
- operands
- "syntactic sugar": symbols used only for readability

The information content of an element of a language, then, depends on the operators and operands, some function of which should lead to a good approximation to our ideal partitioning. This is simply a more formal description of the approach employed by [Bulut 1973].

In his study of student FORTRAN programs, Bulut counted the basic software science [Halstead 1972, 1977] parameters:

- \( \gamma_1 \): the number of unique operators
- \( \gamma_2 \): the number of unique operands
- \( N_1 \): the total number of occurrences of operators
- \( N_2 \): the total number of occurrences of operands

He noted that "the probability of using \( \gamma_1 \) and \( \gamma_2 \) symbols exactly \( N_1 \) and \( N_2 \) times in two different...[expressions] is very slim." Plagiarized copies were found by hand checking programs with identical \( \gamma_1 \), \( \gamma_2 \), \( N_1 \), and \( N_2 \) values. Bulut observed that, as with the length function above, the results of this method are not affected by changes to operand names since such changes will not modify \( \gamma_2 \) or \( N_2 \).

A program to count these four parameters for FORTRAN modules was written [Ottenstein 1976] and used to confirm Bulut's work. Table 1 shows the partitioning imposed on 47 student programs from CS 210 at Purdue University by the 'software science method'. In the formalism developed here, we consider this method a mapping of programs into 4-tuples, \( (\gamma_1, \gamma_2, N_1, N_2) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \), where \( \mathbb{N} \) denotes the set of natural numbers. Two partitions (A and B) have two programs in them; the rest have one. One program in partition B is a copy of the other, with slightly different comments and margining. The other pair is not as immediately detectable as being plagiarized because one author apparently changed all of the variable names and label numbers. Other programs with close correspondence of the parameters were
compared, but without positive results. Thus it seems that a good partitioning was obtained. Copies of the programs in partition A are included in Appendix A with the parameter counts.

The size of a program (in tokens) is given as column N in Table 1. Since $N=N_1+N_2$, the partitions created by the length function mentioned above are supersets of those created by the software science method. Here, the length function creates 10 partitions of size greater than one, while the software science method seems to have given us the ideal partitioning. So at least in this case, the additional information provided by $\gamma_1$ and $\gamma_2$ is well worth the small effort required to obtain it.

Bulut called the chances of two student programs having equal 4-tuples "slim". We can get a more quantitative probability estimate by observing that $1_1, 1_2, N_1$, and $N_2$ all appear to have somewhat normal distributions, in agreement with our intuition. (Appendix B gives the histograms for the four parameters.) In our particular sample, we have:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mean</th>
<th>Median</th>
<th>Mode</th>
<th>S.D.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma_1$</td>
<td>17.00</td>
<td>17</td>
<td>15</td>
<td>2.07</td>
<td>13</td>
<td>24</td>
</tr>
<tr>
<td>$\gamma_2$</td>
<td>35.38</td>
<td>35</td>
<td>35</td>
<td>3.93</td>
<td>27</td>
<td>45</td>
</tr>
<tr>
<td>$N_1$</td>
<td>145.77</td>
<td>135</td>
<td>135</td>
<td>19.30</td>
<td>116</td>
<td>195</td>
</tr>
<tr>
<td>$N_2$</td>
<td>111.36</td>
<td>106</td>
<td>101</td>
<td>17.26</td>
<td>84</td>
<td>154</td>
</tr>
</tbody>
</table>

Assuming this normal distribution, there is clearly a greater likelihood of finding a pair of independently written programs with equal parameter values near the means as there is of finding such a pair with values on the tails. Thus, we can be more confident of a partition's accuracy as its individual parameter values approach the tails of their distribution curves.

Since the four parameters are not mutually independent, we use a multivariate normal density function, $g$, determined by the means vector, $\mu=(17.00, 35.38, 145.77, 111.36)$, and the covariance matrix $[C] = C_{ij} = \text{cov}(X_i - \mu_i, X_j - \mu_j)$ with $X=(\gamma_1, \gamma_2, N_1, N_2)$, to get a feel for the closeness of a 4-tuple to the means vector. The expression $g(X)/g(\mu)$ is 1 at $X=\mu$ and approaches 0 as we move away from the mean. Evaluated at the 4-tuple $X$ for partitions $A$ and $B$, this expression results in 0.45 and 0.018, respectively. This indicates that the programs in partition $B$ are very probably plagiarized (the partition is accurate), while those in $A$ are less probably so. Visual inspection of the programs is clearly warranted in any case, but one would be particularly suspicious of those in partition $B$. Since the accuracy of the partitions varies according to the location of the 4-tuples in the distribution space, it would seem advantageous to find a partitioning function whose range has a constant distribution. The existence of such a function is not
known at present, although one would expect that if such a function were found, it would not be particularly accurate. In general, meaningful measurements of human behaviour produce uneven distributions.

Many alterations made by students to copied programs will be transparent to this method. Cosmetic transformations such as the reordering of time-independent statements, reformatting of text, and renaming variables and labels will have no effect at all on $n_1$, $n_2$, $N_1$, or $N_2$. Most non-cosmetic alterations fall into one of six well-defined impurity classes, all of which are detectable by a slightly more sophisticated counter. Unfortunately, a student who cheated on only part of a program will not be detected.

Since the parameter counting routine was developed for other purposes, its $300 or so development cost is not significant here: its running cost is about five cents ($0.05) per 100-line student program on a CDC 6500. (This would be less were it not that the routine was written in ANSI-FORTRAN for portability and self-analysis.) Thus, this method of detecting plagiarism is both inexpensive and rapid. The preventive element mentioned in the title is simply the deterrent created by making it difficult to cheat successfully.

It seems that this method can not only be applied to programs in other computer languages, but to any assignment which requires the submission of written material. Of course, programs are the only practical item for measurement since they are already in machine-readable form, but software science has been applied with some success to English [Kuhn 1975, Halstead 1977] and one might hypothesize that similar results can be obtained there.

ACKNOWLEDGEMENTS

Special notes of thanks are due Dwight Andrews of Purdue University for collecting copies of his students' programs expressly for this study and to Professor Halstead, also at Purdue, for his encouragement and insights into software science.

1The impurity classes are [Bulut 1974]:
(1) self-cancelling operations
(2) ambiguous usage of an operand
(3) synonymous usages of operands
(4) common subexpressions
(5) unnecessary replacements
(6) unfactored expressions
Table 1: 47 student program parameter values as partitioned by the software science method (left) and the length function (right).
APPENDIX A

Source Listings of Programs in Partition B
1. N=1
2. IM=0
3. IF1=0
4. IP1=0
5. IG2=0
6. IF2=0
7. IP2=0
8. C HAVE REFERENCED THE COUNTERS
9. READ 111, KQTS, KFTS, KPTS
10. 111 FORMAT (12, X, 12, X, 12)
11. READ 120, MAX
12. 120 FORMAT (I2)
13. C HAVE READ QUANTITIES ON HAND AND HEADER NUMBER
14. 10 READ 100, NUM, IC0DE, IKQTS, IKFTS, IKPTS
15. 100 FORMAT (11, X, I4, X, I2/X, I2; I2)
16. C HAVE READ A DATA CARD. THE THREE SUCCEEDING IF STATEMENTS
17. C CHECK ORDER QUANTITIES AGAINST QUANTITIES ON HAND
18. C 'INSUFFICIENT QUANTITY' RECEIPT PRINTED IF APPLICABLE
19. IF (IKQTS. LE. KQTS) GO TO 20
20. PRINT 200, NUM, IC0DE
21. PRINT 205
22. PRINT 300
23. GO TO 44
24. 20 IF (IKFTS. LE. KFT5) GO TO 30
25. PRINT 200, NUM, IC0DE
26. PRINT 205
27. PRINT 300
28. GO TO 44
29. 30 IF (IKPTS. LE. KPTS) GO TO 40
30. PRINT 200, NUM, IC0DE
31. PRINT 205
32. PRINT 300
33. GO TO 44
34. C IF ORDER CAN BE FILLED, COSTS ARE COMPUTED
35. C AND A RECEIPT PRINTED
36. 40 KQTS=KQTS-IOQTS
37. KFTS=KFTS-IOFTS
38. KPTS=KPTS-IOPTS
39. QC0ST=6. 05*FLOAT(IOQTS)
40. FC0ST=4. 15*FLOAT(IOFTS)
41. PC0ST=2. 25*FLOAT(IOPTS)
42. 1BT=QC0ST+FC0ST+PC0ST
43. IF (NUM. EQ. 1) GO TO 66
44. PRINT 200, NUM, IC0DE
45. GO TO 77
46. 66 PRINT 201, NUM, IC0DE
47. 77 PRINT 210
48. PRINT 220, KQTS, QC0ST
49. PRINT 230, IOQTS, FC0ST
50. PRINT 240, IOPTS, PC0ST
51. PRINT 250, 1BT
52. PRINT 300
53. C AFTER THE RECEIPT IS PRINTED, THE COSTS FOR EACH STORE ARE
54. C UPDATED TO BE RECALLED AS A SUMMARY WHEN ALL CARDS ARE READ.
55. C SUMMARY VARIABLES HAVE APPROPRIATE SUFFICES.
56. C 1 FOR STORE NUMBER 1 AND 2 FOR STORE NUMBER 2
57. IF(NUM. EQ. 2) GO TO 33
58. 102=102+100TS
59. IF2=IF2+IFPTS
60. IP2=IP2+IPPTS
61. QCS2=6.05*FL0RT(IF2)
62. PC052=2.25*FL0RT(IP2)
63. GT0T2=QCS2+PC052+PC052
64. G0 T0 44
65. JI IF1=IF1+IFPTS
66. IP1=IP1+IPPTS
67. QCS1=6.05*FL0RT(IF1)
68. PC051=2.25*FL0RT(IP1)
69. GT0T1=QCS1+PC051+PC051
70. N=N+1
71. THE NEXT STEP CHECKS THE CARD COUNT AGAINST THE HEADER
72. IFN LE NMAX G0 T0 10
73. PRINT 260
74. PRINT 210
75. PRINT 200.101, QCS1
76. PRINT 200.102, QCS2
77. PRINT 200.101, PC051
78. PRINT 200.102, PC052
79. PRINT 200.101, FC051
80. PRINT 200.102, FC052
81. PRINT 200.101, UT0T1
82. PRINT 300
83. PRINT 290
84. PRINT 280
85. PRINT 210
86. PRINT 200.101, QCS1
87. PRINT 200.102, QCS2
88. PRINT 200.101, PC051
89. PRINT 200.102, PC052
90. PRINT 200.101, FC051
91. PRINT 200.102, FC052
92. 200 FORMAT ('0.15X,'STORE ',.11.3X,'ORDER CODE ',.14)
93. 201 FORMAT ('1.15X,'STORE ',.11.3X,'ORDER CODE ',.14)
94. 210 FORMAT ('0.15X,'ITEM ',.3X,'PRICE ',.5X,'QST' )
95. 211 FORMAT ('1.15X,'ITEM ',.3X,'PRICE ',.5X,'QST' )
96. 220 FORMAT ('0.12X.12X,'QUART(S) # 6.05 $',F6.2)
97. 221 FORMAT ('1.12X.12X,'QUART(S) # 6.05 $',F6.2)
98. 230 FORMAT ('0.12X.12X,'QUART(S) # 2.25 $',F6.2)
99. 231 FORMAT ('1.12X.12X,'QUART(S) # 2.25 $',F6.2)
100. 240 FORMAT ('0.27X,'TOTAL ',.7X,F7.2)
101. 241 FORMAT ('1.17X,'TOTAL ',.7X,F7.2)
102. 250 FORMAT ('0.21X,'GRAND TOTAL ',.7X,F7.2)
103. 260 FORMAT ('0.17X,'STORE ',.2X,'TOTAL BILL')
104. 270 FORMAT ('0.17X,'STORE ',.2X,'TOTAL BILL')
105. 280 FORMAT ('0.17X,'STORE ',.2X,'TOTAL BILL')
106. 290 FORMAT ('0.17X,'STORE ',.2X,'TOTAL BILL')
107. STOP
108. END
STATISTICS FOR THIS MODULE:

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>FREQUENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.O.S.</td>
<td>40</td>
</tr>
<tr>
<td>() OR DO</td>
<td>9</td>
</tr>
<tr>
<td>IF</td>
<td>6</td>
</tr>
<tr>
<td>+</td>
<td>9</td>
</tr>
<tr>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>=</td>
<td>29</td>
</tr>
<tr>
<td>LE</td>
<td>4</td>
</tr>
<tr>
<td>END</td>
<td>2</td>
</tr>
<tr>
<td>GOTO 20</td>
<td>1</td>
</tr>
<tr>
<td>GOT0 44</td>
<td>4</td>
</tr>
<tr>
<td>GOT0 30</td>
<td>1</td>
</tr>
<tr>
<td>GOT0 40</td>
<td>1</td>
</tr>
<tr>
<td>GOT0 66</td>
<td>1</td>
</tr>
<tr>
<td>GOT0 77</td>
<td>1</td>
</tr>
<tr>
<td>GOT0 33</td>
<td>1</td>
</tr>
<tr>
<td>GOT0 10</td>
<td>1</td>
</tr>
<tr>
<td>FLMT</td>
<td>9</td>
</tr>
</tbody>
</table>

\[\text{ETA1} = 18\]
\[\text{NI} = 135\]

<table>
<thead>
<tr>
<th>OPERAND</th>
<th>FREQUENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>1</td>
</tr>
<tr>
<td>KPTS</td>
<td>3</td>
</tr>
<tr>
<td>QCBST</td>
<td>2</td>
</tr>
<tr>
<td>QCBS1</td>
<td>2</td>
</tr>
<tr>
<td>QCBS2</td>
<td>2</td>
</tr>
<tr>
<td>PCSR</td>
<td>2</td>
</tr>
<tr>
<td>PCBS1</td>
<td>2</td>
</tr>
<tr>
<td>PCBS2</td>
<td>2</td>
</tr>
<tr>
<td>IB0TS</td>
<td>3</td>
</tr>
<tr>
<td>NUM</td>
<td>1</td>
</tr>
<tr>
<td>GT0T1</td>
<td>1</td>
</tr>
<tr>
<td>GT0T2</td>
<td>1</td>
</tr>
<tr>
<td>IB0TS</td>
<td>5</td>
</tr>
<tr>
<td>IB0TS</td>
<td>5</td>
</tr>
<tr>
<td>IP1</td>
<td>4</td>
</tr>
<tr>
<td>IP2</td>
<td>4</td>
</tr>
<tr>
<td>FC0ST</td>
<td>2</td>
</tr>
<tr>
<td>FC0S1</td>
<td>2</td>
</tr>
<tr>
<td>FC0S2</td>
<td>2</td>
</tr>
<tr>
<td>IF1</td>
<td>4</td>
</tr>
<tr>
<td>IF2</td>
<td>4</td>
</tr>
<tr>
<td>K0S1</td>
<td>4</td>
</tr>
<tr>
<td>K0S2</td>
<td>2</td>
</tr>
<tr>
<td>K0S3</td>
<td>2</td>
</tr>
<tr>
<td>K0S4</td>
<td>2</td>
</tr>
<tr>
<td>M0X</td>
<td>1</td>
</tr>
<tr>
<td>K0TS</td>
<td>3</td>
</tr>
<tr>
<td>I0EK</td>
<td>4</td>
</tr>
<tr>
<td>0.00E+00</td>
<td>1</td>
</tr>
<tr>
<td>0.00E+00</td>
<td>6</td>
</tr>
<tr>
<td>6.00E+00</td>
<td>3</td>
</tr>
<tr>
<td>4.10E+00</td>
<td>3</td>
</tr>
<tr>
<td>2.00E+00</td>
<td>3</td>
</tr>
</tbody>
</table>

\[\text{ETA2} = 32\]
\[\text{NI} = 95\]
1. PROGRAM 2 CS 210
2. C FOLLOWING CALCULATIONS ARE FOR D.T.W.P. PERTAINING TO WEEKLY SALES
3. N=1
4. IQTS=0
5. IF=0
6. IPTS=0
7. IQT=0
8. IFIB=0
9. IPT=0
10. IQTB=0
11. READ333, LOTS, LFTS, LPTS
12. FORMAT(12, X, 12, X, 12)
13. READ330, NUMST
14. FORMAT(12)
15. IF=0, IQTB, IQTB, IPTB, IPTB
16. FORMAT(14, X, 12, X, 12, X, 12)
17. C THIS DETERMINES WHETHER OR NOT THE ORDER CAN BE FILLED.
18. IF IQTB .LE. LOTS GOTO 11
19. C IF THE ORDER CANNOT BE FILLED, THIS INFORMATION WILL BE PRINTED.
20. PRINT100, NUMST, IQTB
21. PRINT110
22. GOTO55
23. IF IFIB .LE. LFTS GOTO12
24. PRINT100, NUMST, IFIB
25. PRINT110
26. GOTO55
27. IF IPTB .LE. LPTS GOTO20
28. PRINT100, NUMST, IPTB
29. PRINT110
30. GOTO55
31. LOTS=LOTS-IQT
32. LFTS=LFTS-IFIB
33. LPTS=LPTS-IPTB
34. C FOLLOWING DETERMINES ALL COST INFORMATION IF ORDER CAN BE FILLED.
35. QCOST=6.05*FLOAT(IQT)
36. FCOST=4.15*FLOAT(IFIB)
37. PCOST=2.25*FLOAT(IPTB)
38. TOT-QCOST+FCOST+PCOST
39. IF NUMST .EQ. 1 GOTO 77
40. PRINT100, NUMST, IQTB
41. PRINT110
42. THIS PRINTS OUT STORE ORDERS
43. PRINT111
44. PRINT112, IPTB, QCOST
45. PRINT113, IFIB, FCOST
46. PRINT114, IPTB, FCOST
47. PRINT115, TOT
48. PRINT119
49. GOTO25
50. IF NUMST .EQ. 1 GOTO25
51. GOTO107
52. IF IFIB .LE. IPTB GOTO52
53. IF IPTB .LE. IPTB GOTO54
54. QCO5=6.05*FLOAT(IQT)
55. FCOST=4.15*FLOAT(IFIB)
56. PCOST=2.25*FLOAT(IPTB)
57. TOT=QOST+FCOST+FCOST
58. GOTO55
59. 25 IOTS=IOTS+IOTS
60. IFIF=IFIF+IFIF
61. IPTS=IPTS+IPTS
62. QCO50=6.05*FLOAT(IOTS)
63. F050=4.15*FLOAT(IFIF)
64. P050=2.25*FLOAT(IPTS)
65. GT0TO=QCO50+F050+P050
66. 59 H=M+1
67. IF(IOTS.LE.F050).GOT010
68. C. THIS PRINTS OUT THE TOTAL BILL.
69. PRINT116
70. PRINT11
71. PRINT112. IOTS, QCO50
72. PRINT113. IFIF, F050
73. PRINT114. IPTS, P050
74. PRINT117. GT0TO
75. PRINT119
76. PRINT11
77. PRINT111
78. PRINT112. IOTS, QCO50
79. PRINT113. IFIF, F050
80. PRINT114. IPTS, P050
81. PRINT117. GT0TO
82. 100 FORMAT('O', 12X, 'STORE:', X, 11.2X, 'ORDER CODE:', X, 14)
83. 101 FORMAT('O', 12X, 'STORE:', X, 11.2X, 'ORDER CODE:', X, 14)
84. 110 FORMAT('O', 11X, 'ORDER NOT FILLED. ')
85. 1 INSUFFICIENT STOCK ON HAND ****'
86. 111 FORMAT('O', 12X, 'ITEM: 5X: 'PRICE': 6X: 'CST')
87. 112 FORMAT('O', 12X, 'QUART(S)': 6X: 0.2)
88. 113 FORMAT('O', 12X, 'FIFTH(S)': 4.15 1.56: 0.2)
89. 114 FORMAT('O', 12X, 'PINT(S)': 2.25 1.45: 0.2)
90. 115 FORMAT('O', 27X, 'TOTAL 1. F7.2)
91. 116 FORMAT('O', 15X, 'STORE 1 TOTAL BILL')
92. 117 FORMAT('O', 15X, 'STORE 2 TOTAL BILL')
93. 118 FORMAT('O', 15X, 'GRAND TOTAL 1. F7.2)
94. 119 FORMAT('O')
95. STOP
96. END
### Statistics for This Module:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>E O S</td>
<td>40</td>
</tr>
<tr>
<td>O NW DO</td>
<td>9</td>
</tr>
<tr>
<td>IF</td>
<td>6</td>
</tr>
<tr>
<td>=</td>
<td>2</td>
</tr>
<tr>
<td>&gt;</td>
<td>12</td>
</tr>
<tr>
<td>LE</td>
<td>4</td>
</tr>
<tr>
<td>EQ</td>
<td>2</td>
</tr>
<tr>
<td>GOTO 11</td>
<td>1</td>
</tr>
<tr>
<td>GOTO 55</td>
<td>4</td>
</tr>
<tr>
<td>GOTO 12</td>
<td>1</td>
</tr>
<tr>
<td>GOTO 20</td>
<td>1</td>
</tr>
<tr>
<td>GOTO 77</td>
<td>1</td>
</tr>
<tr>
<td>GOTO 22</td>
<td>1</td>
</tr>
<tr>
<td>GOTO 25</td>
<td>1</td>
</tr>
<tr>
<td>POTA 30</td>
<td>1</td>
</tr>
</tbody>
</table>

### Operand Frequency

<table>
<thead>
<tr>
<th>Operand</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOT</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>5</td>
</tr>
<tr>
<td>OPO5O</td>
<td>3</td>
</tr>
<tr>
<td>OPA0I</td>
<td>2</td>
</tr>
<tr>
<td>PEMS5O</td>
<td>2</td>
</tr>
<tr>
<td>PK5ST</td>
<td>2</td>
</tr>
<tr>
<td>GT5O</td>
<td>1</td>
</tr>
<tr>
<td>NMAK</td>
<td>1</td>
</tr>
<tr>
<td>GOTO5</td>
<td>5</td>
</tr>
<tr>
<td>GOTO5O</td>
<td>1</td>
</tr>
<tr>
<td>PCD5S</td>
<td>2</td>
</tr>
<tr>
<td>IOT</td>
<td>4</td>
</tr>
<tr>
<td>JTOE</td>
<td>4</td>
</tr>
<tr>
<td>IFID</td>
<td>5</td>
</tr>
</tbody>
</table>

ETR = 12
N2 = 135

ETR2 = 32
N2 = 95
APPENDIX B

Histograms for $\gamma_1$, $\gamma_2$, $N_1$, and $N_2$ for the Observed Sample
DISTRIBUTION OF N2 VALUES

DISTRIBUTION OF ETA2 VALUES
References


