APPENDIX A

Surrogate Oracles, Generalized Dependency and Simpler Models

Larry Wilson
Old Dominion University
Supported by NAG-1-750

Abstract

Software reliability models require the sequence of interfailure times from the debugging process as input. We have previously illustrated that using data from replicated debugging could greatly improve reliability predictions. However, inexpensive replication of the debugging process requires the existence of a cheap, fast error detector. We can design laboratory experiments around a gold version which is used as an oracle or around an n-version error detector. Unfortunately, we can not expect software developers to have an oracle or to bear the expense of n-versions. We are investigating a generic technique for approximating replicated data by using the partially debugged software as a difference detector.

We believe that the failure rate of each fault has significant dependence on the presence or absence of other faults. Thus, in order to discuss a failure rate for a known fault, we need to specify the presence or absence of each of the other known faults.

Also, we are interested in simpler models which use shorter input sequences without sacrificing accuracy. In fact we, conjecture a possible gain in performance.

To investigate these propositions, we are using NASA computers running LIC (RTI) versions to generate data. This data will be used to label the debugging graph associated with each version. These labeled graphs will be used to test the utility of a surrogate oracle, to analyze the dependent nature of fault failure rates and to explore the feasibility of reliability models which use the data of only the most recent failures.

VALUE OF PDG → guides intuition in Reliability Prediction

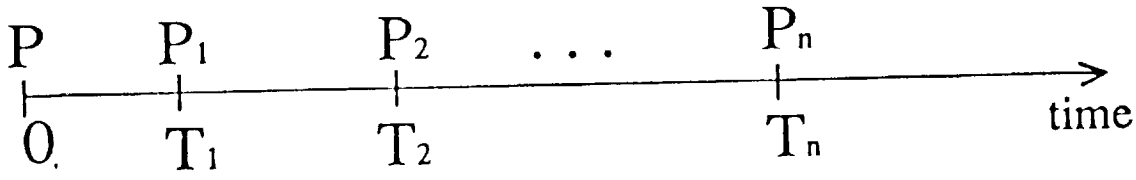
DATA → Real World vs Laboratory

USEFUL — when Replicated

OBTAINABLE → Real World with Surrogate oracle

SURROGATE ORACLE
DATA DEPENDENCE
NEW MODELS

Normal Debugging



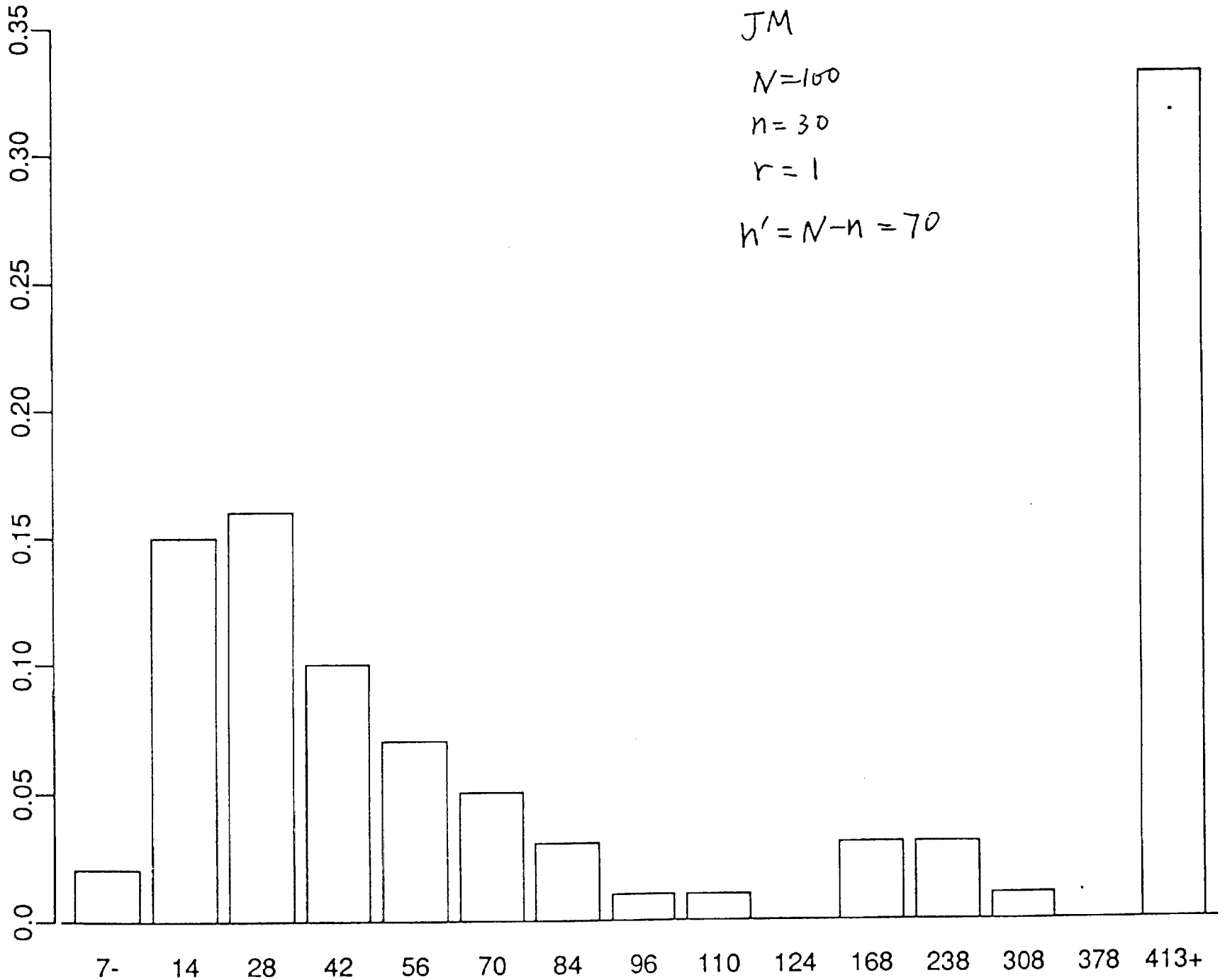
$t_i = T_i - T_{i-1}$ interfailure times

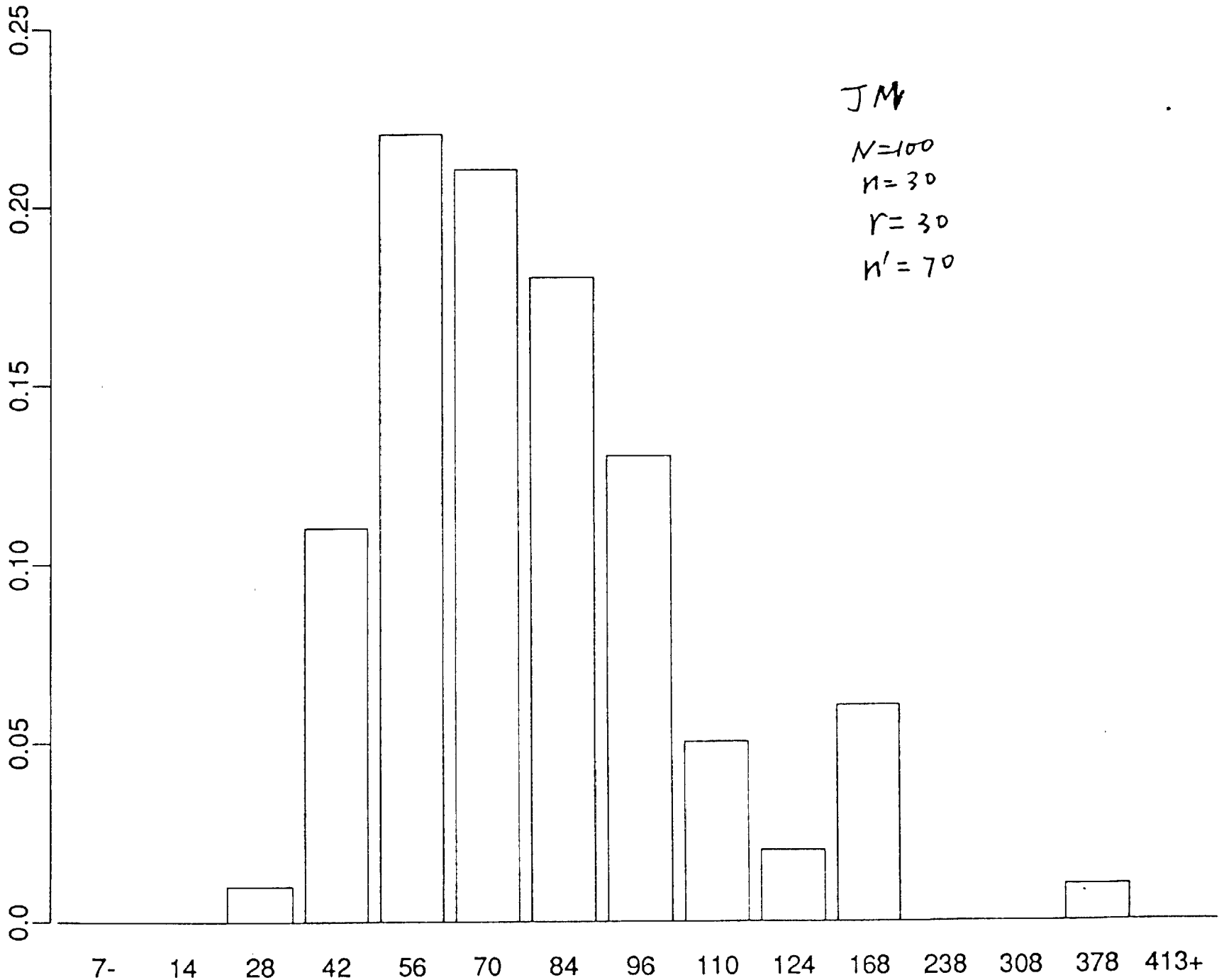
Models take as input (t_1, t_2, . . . , t_n) and produce estimates of λ

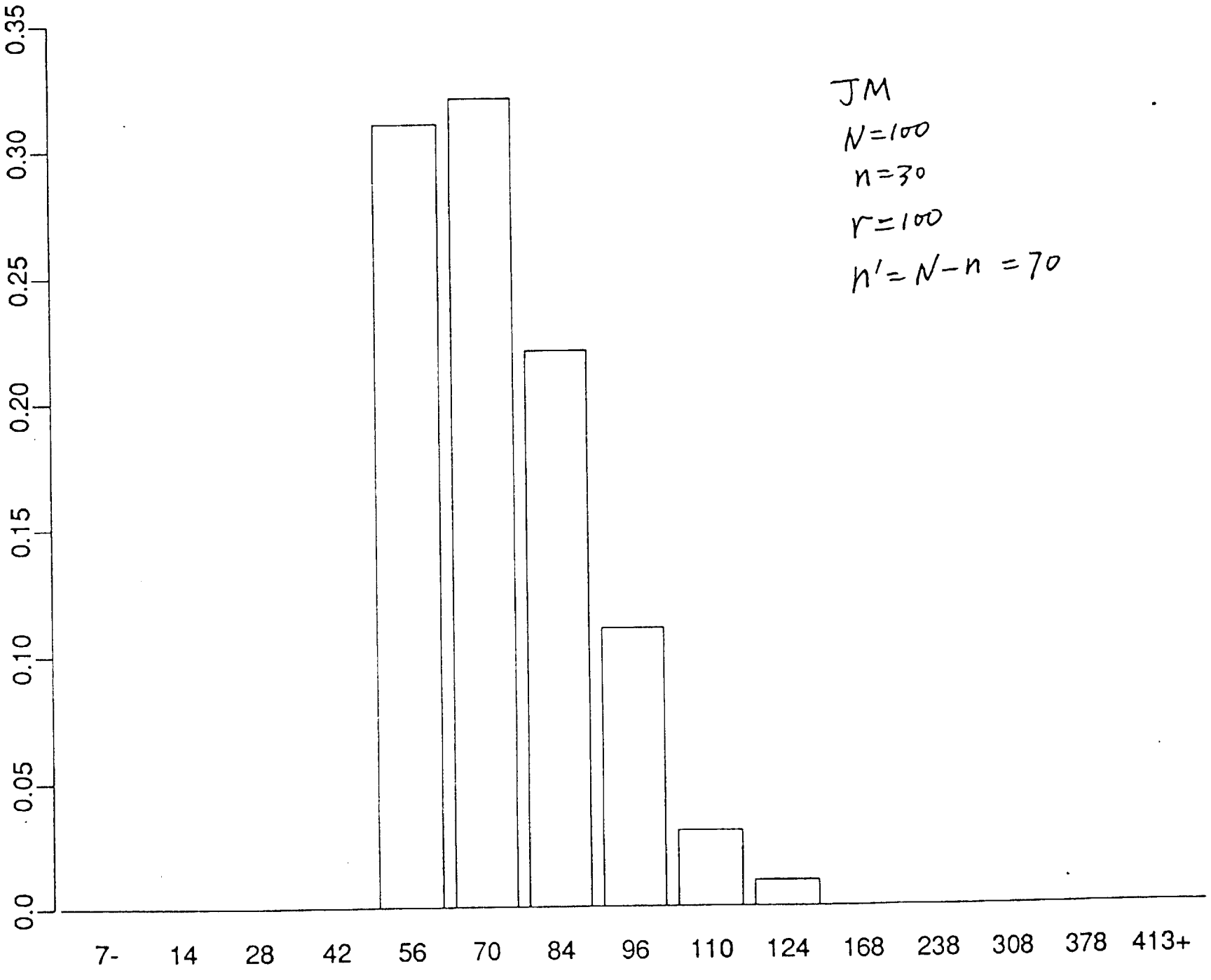
Necessities

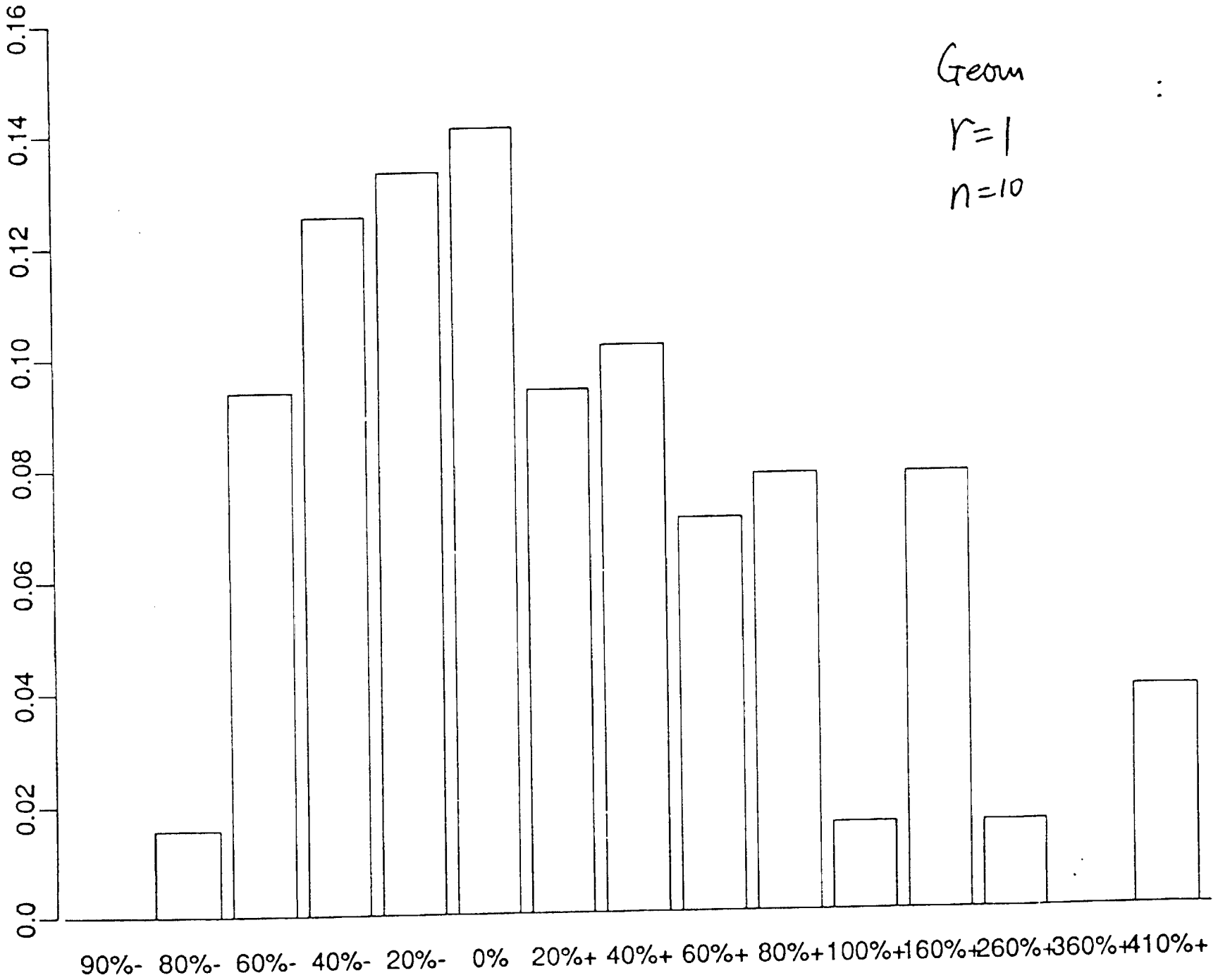
INPUT DISTRIBUTION / RANDOM INPUTS

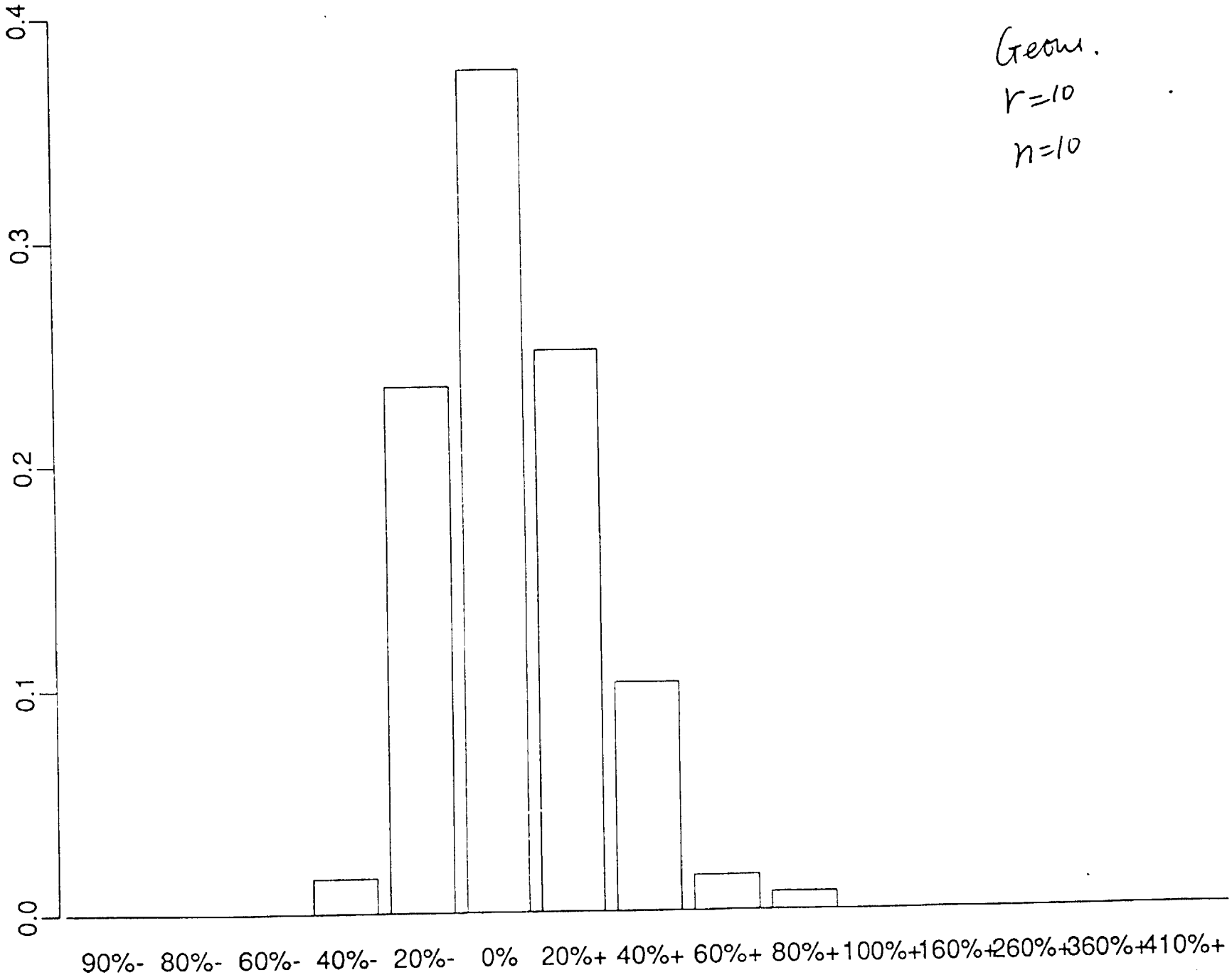
ERROR DETECTOR


JM
N=100
n=30
r=1
h' = N-n = 70
0.0
0.05
0.10
0.15
0.20
0.25
0.30
0.35
7-
14
28
42
56
70
84
96
110
124
168
238
308
378
413+

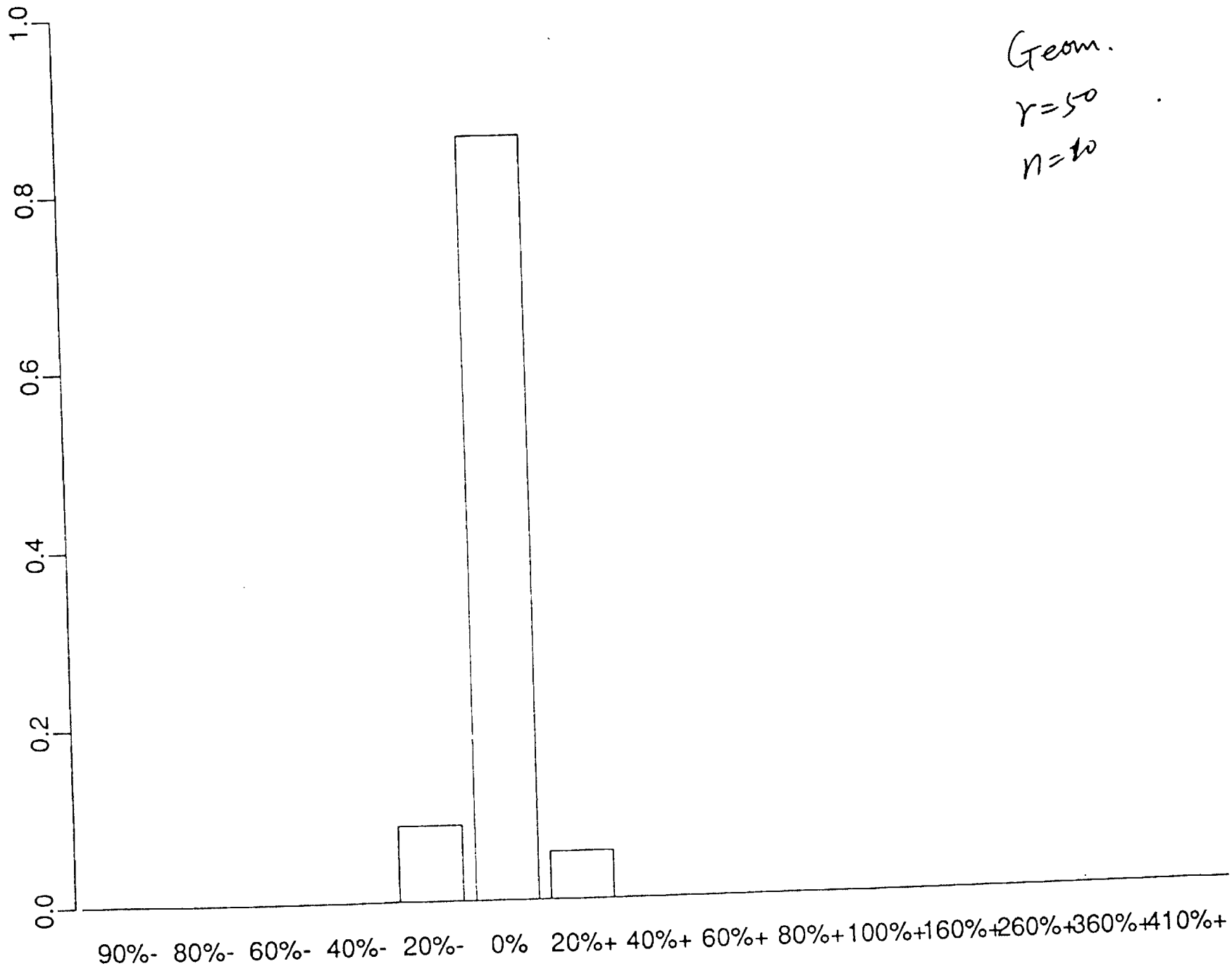

JM
N=100
n=30
r=30
n'=70
0.0
0.05
0.10
0.15
0.20
0.25
7-
14
28
42
56
70
84
96
110
124
168
238
308
378
413+


JM
N=100
n=30
r=100
n'=N-n =70
0.0
0.05
0.10
0.15
0.20
0.25
0.30
0.35
7-
14
28
42
56
70
84
96
110
124
168
238
308
378
413+


Geom
r=1
n=10
0.0
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
90%-
80%-
60%-
40%-
20%-
0%
20%+
40%+
60%+
80%+
100%+
160%+
260%+
360%+
410%+



Geom.
$r=10$
$n=10$


Geom.
r=50
n=10
1.0
0.8
0.6
0.4
0.2
0.0
90%- 80%- 60%- 40%- 20%- 0% 20%+ 40%+ 60%+ 80%+ 100%+ 160%+ 260%+ 360%+ 410%+

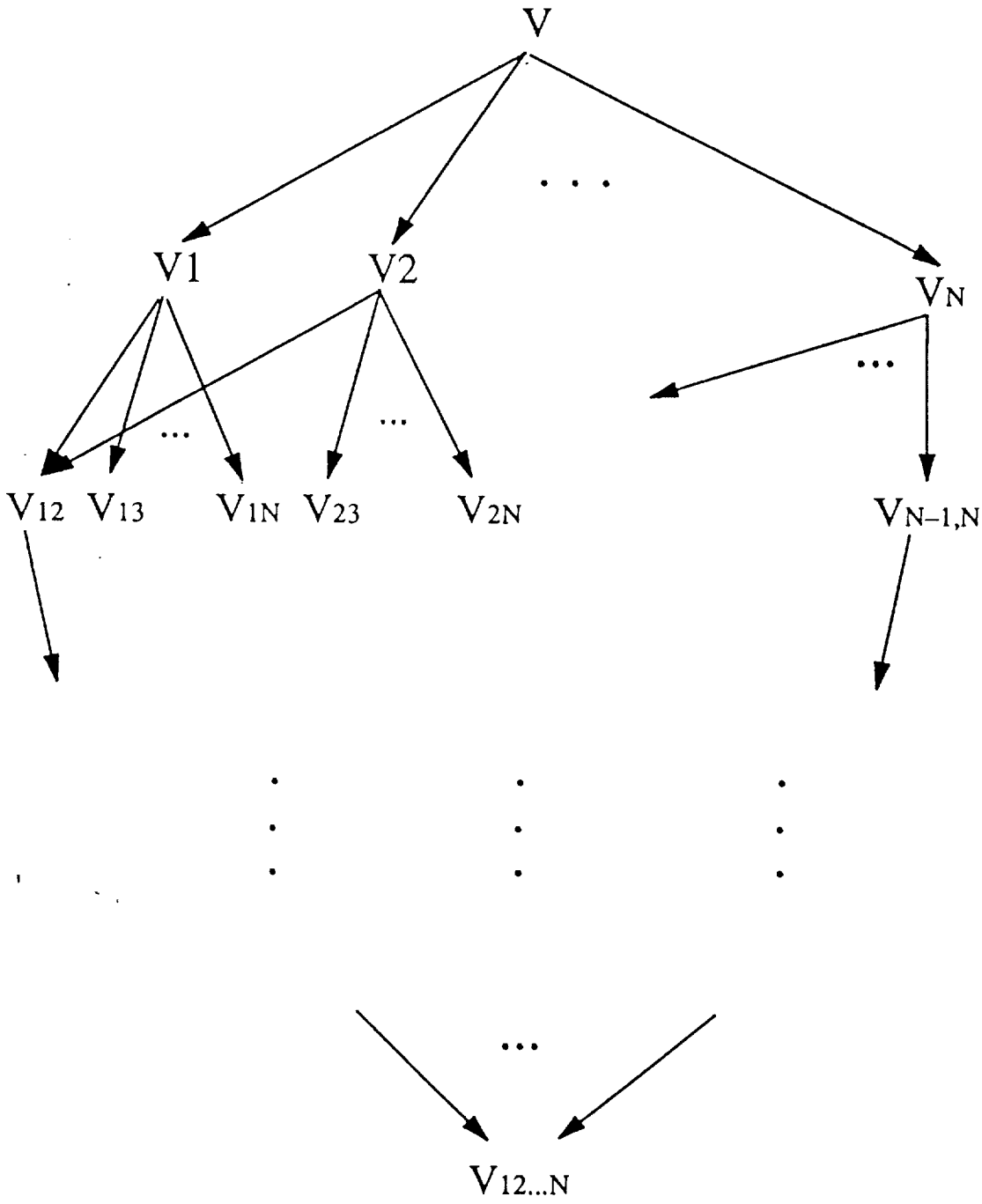
- Conclusion 1: These SR models should never be used to predict software reliability if we only use normal debugging process.

- Conclusion 2: The models are stable after the randomness is removed by replicated debugging. With replication, conceivable to use models.

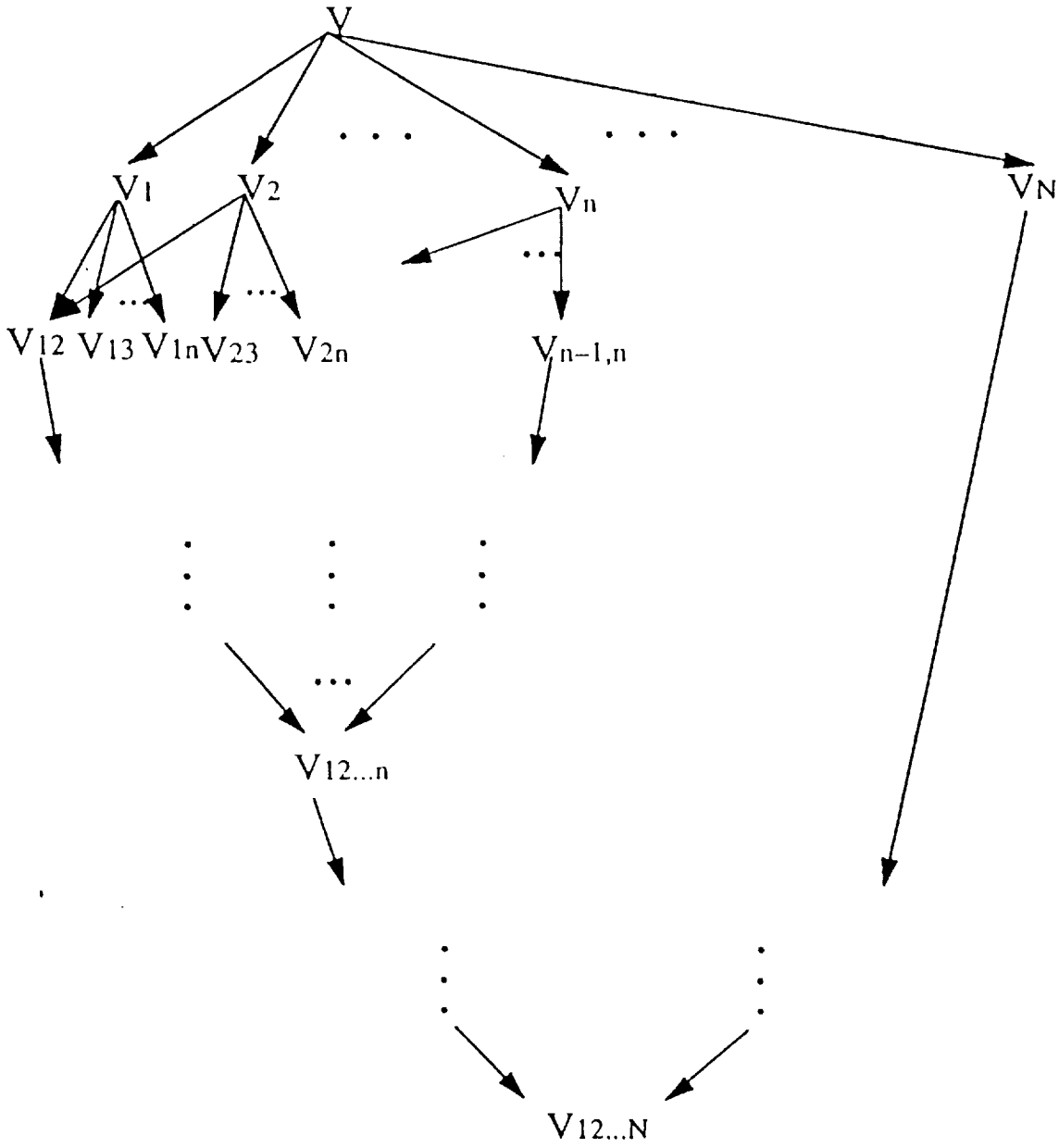
- Future:

1. GCS
2. Front end for repliability models.
3. Estimate and control the cost of replication.
4. Analyze debug graph to get better models.

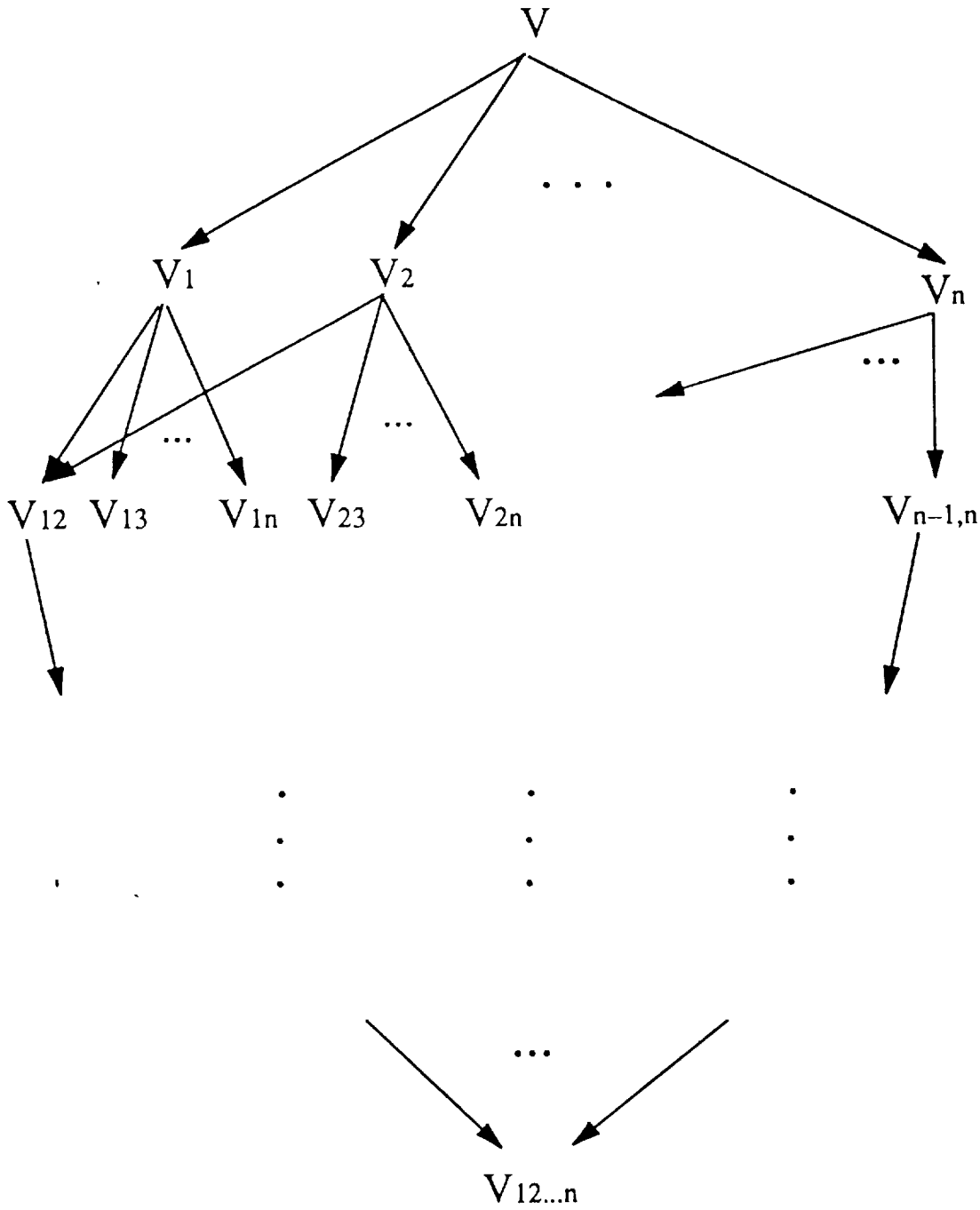
Debug Graph



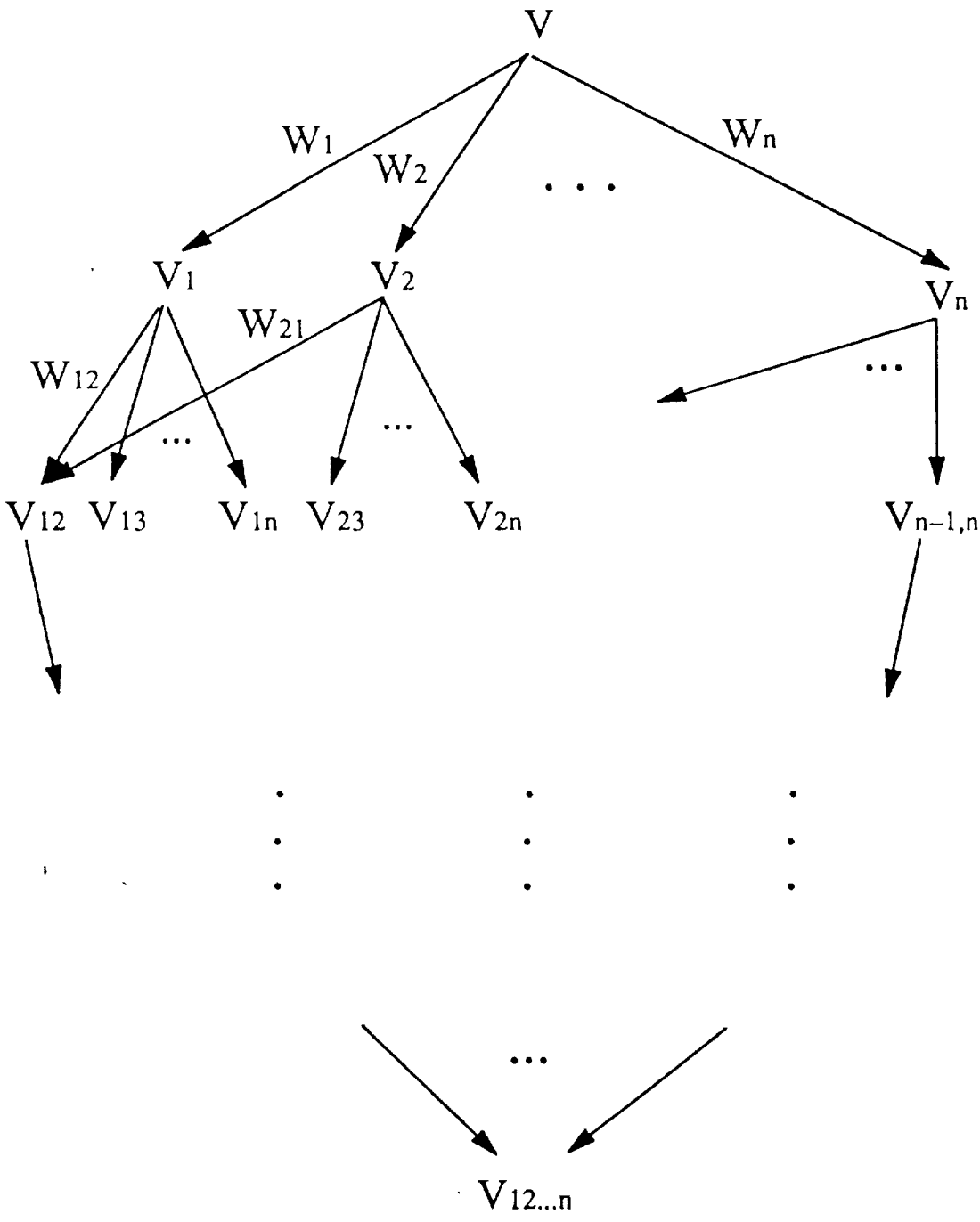
EMBEDED PARTIAL DEBUG GRAPH



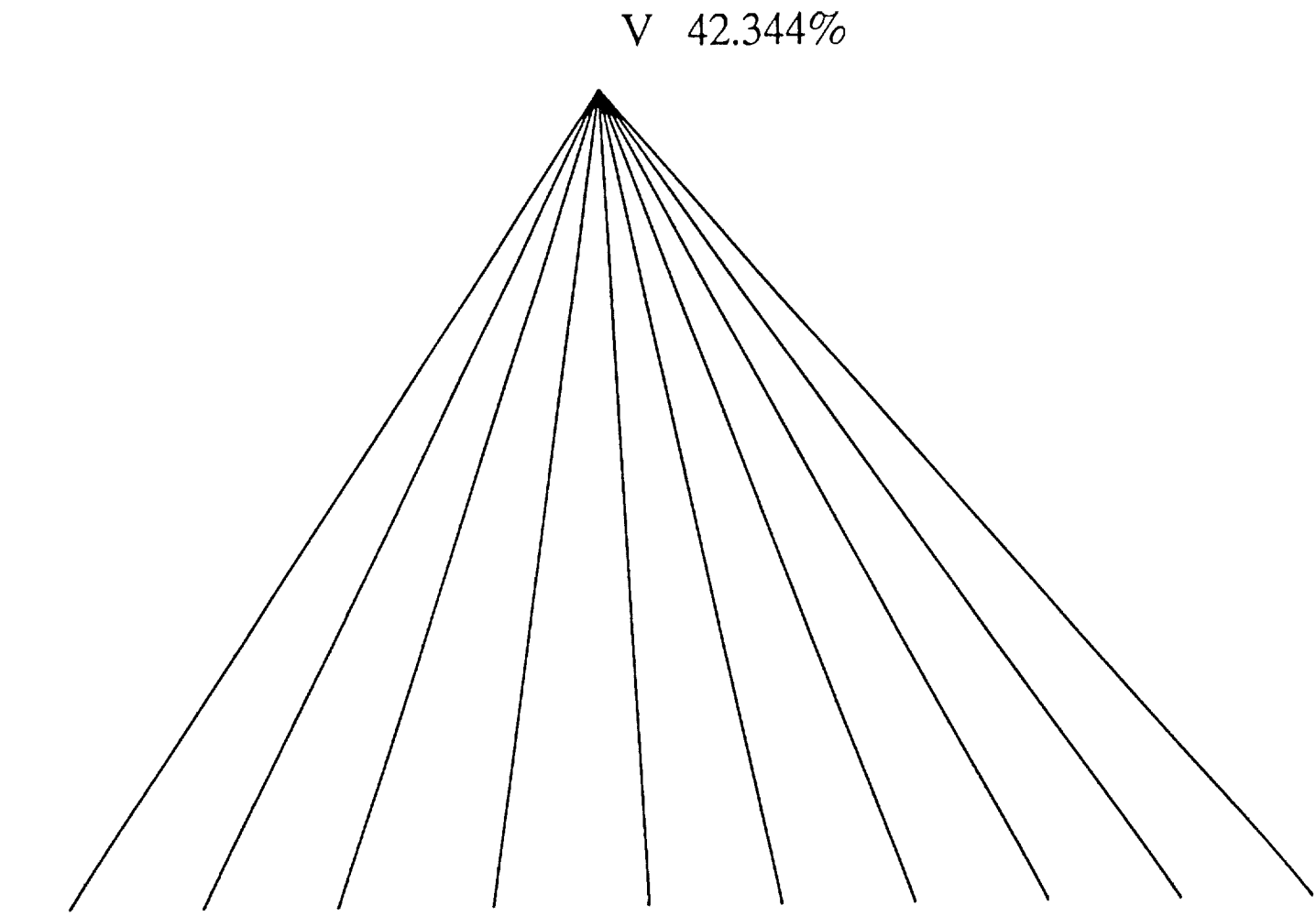
Partial Debug Graph



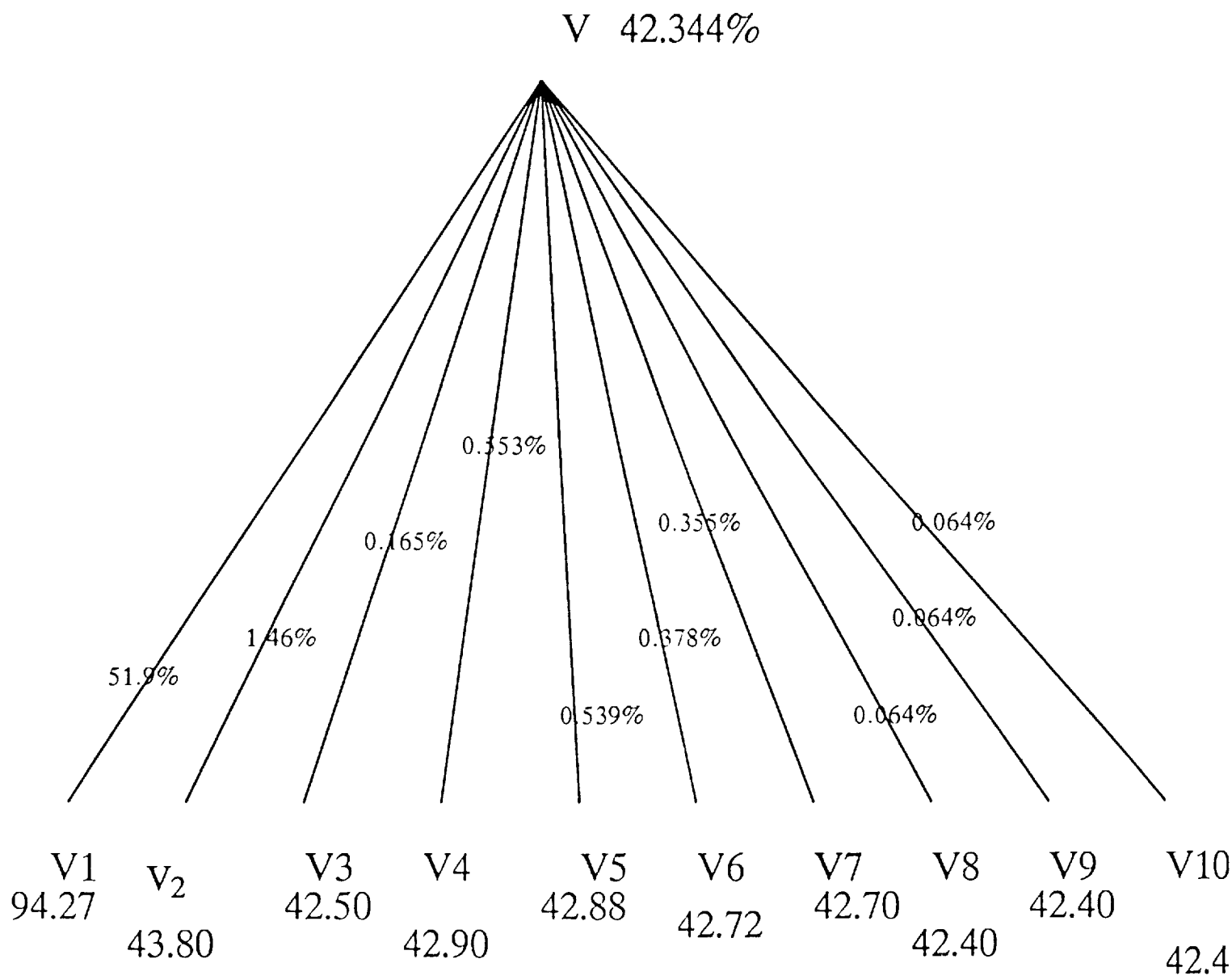
Weighted Partial Debug Graph



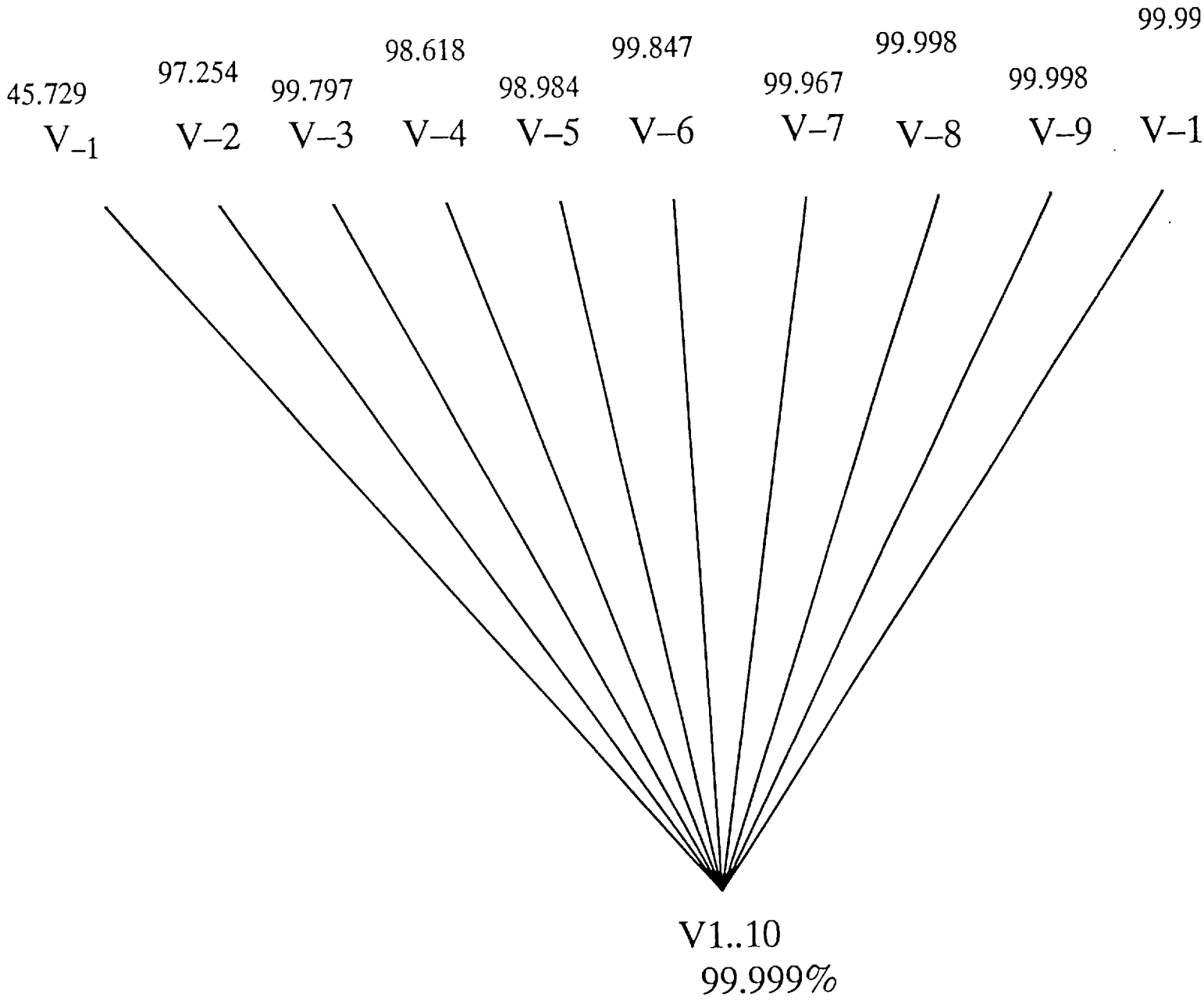
RPDG1


V 42.344%
V1
94.27
V2
43.80
V3
42.50
V4
42.90
V5
42.88
V6
42.72
V7
42.70
V8
42.40
V9
42.40
V10
42.4

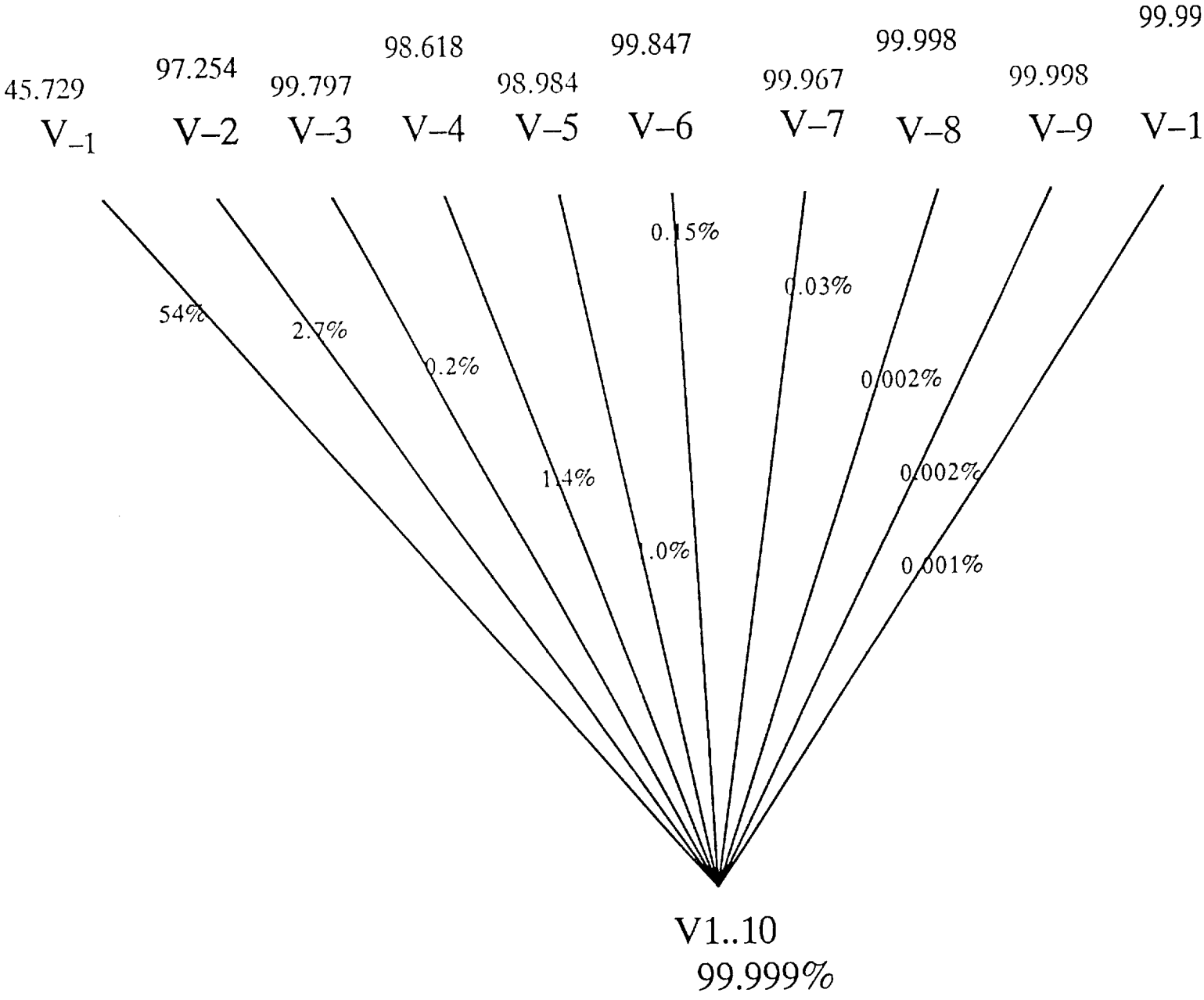
ΔRPDG1

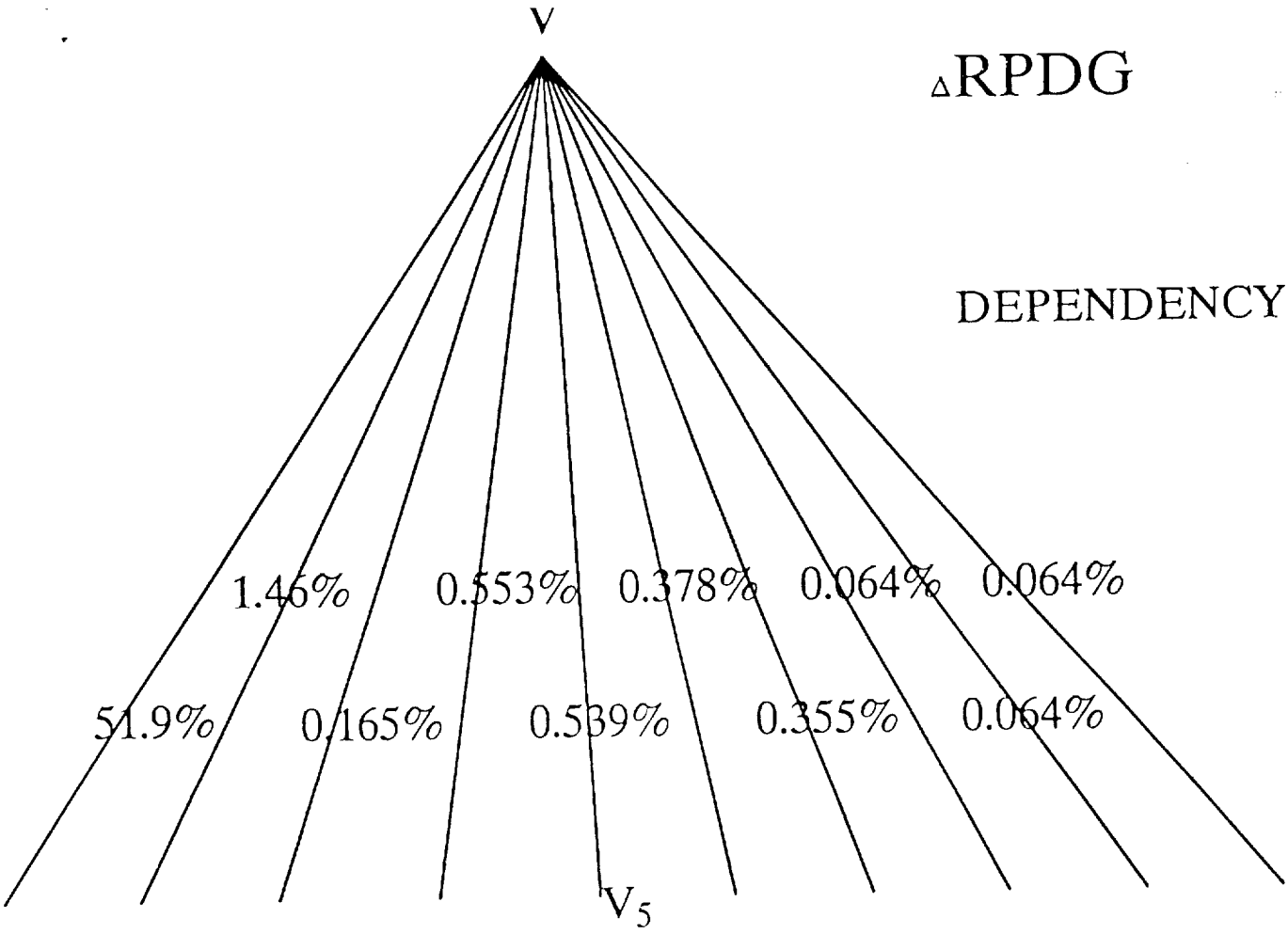

V 42.344%
51.9%
1.46%
0.165%
0.353%
0.539%
0.378%
0.355%
0.064%
0.064%
0.064%
V1
94.27
V2
43.80
V3
42.50
V4
42.90
V5
42.88
V6
42.72
V7
42.70
V8
42.40
V9
42.40
V10
42.4

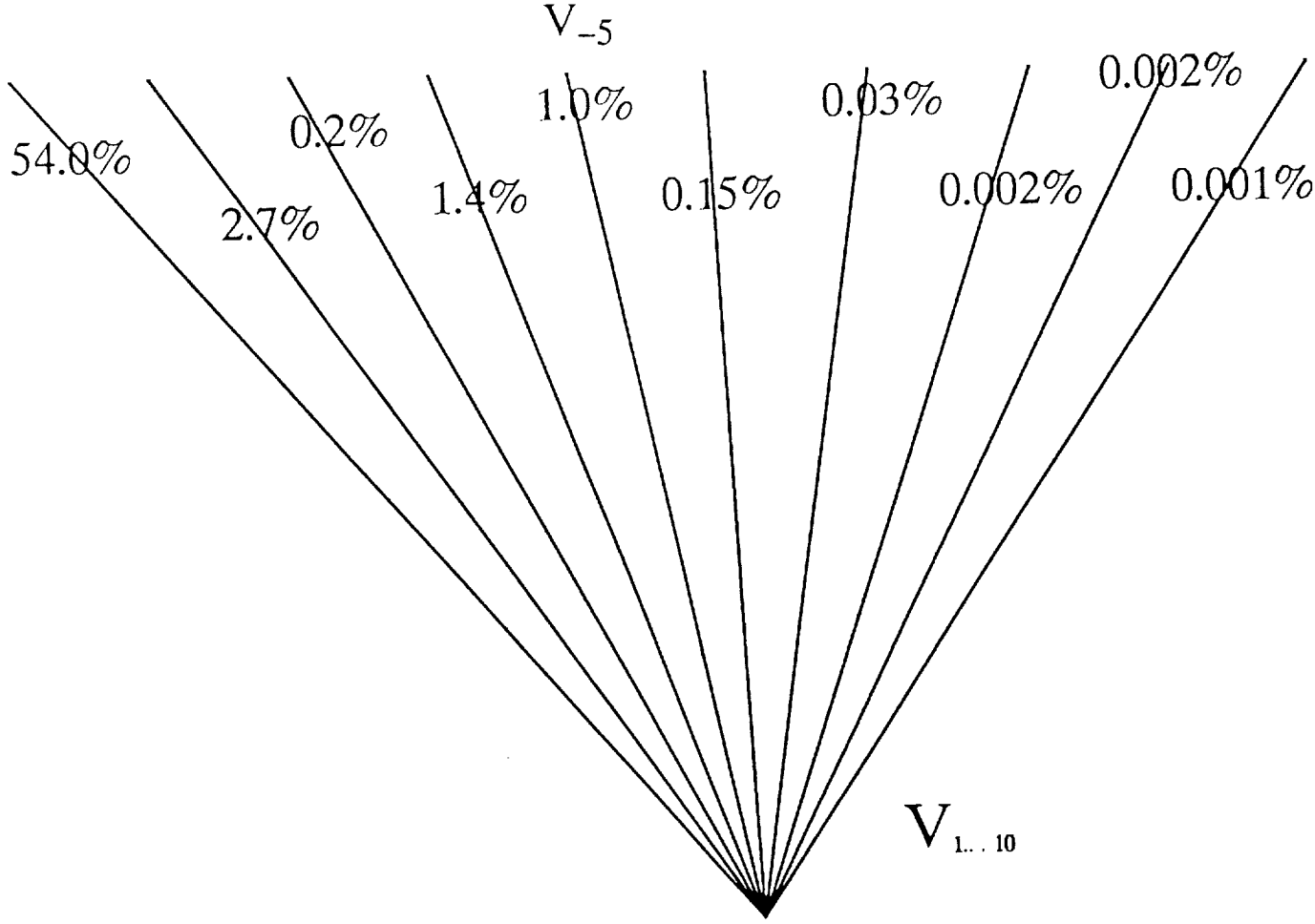
RPDG 9


45.729
V–1
97.254
V–2
99.797
V–3
98.618
V–4
98.984
V–5
99.847
V–6
99.967
V–7
99.998
V–8
99.998
V–9
99.99
V–1
V1..10
99.999%

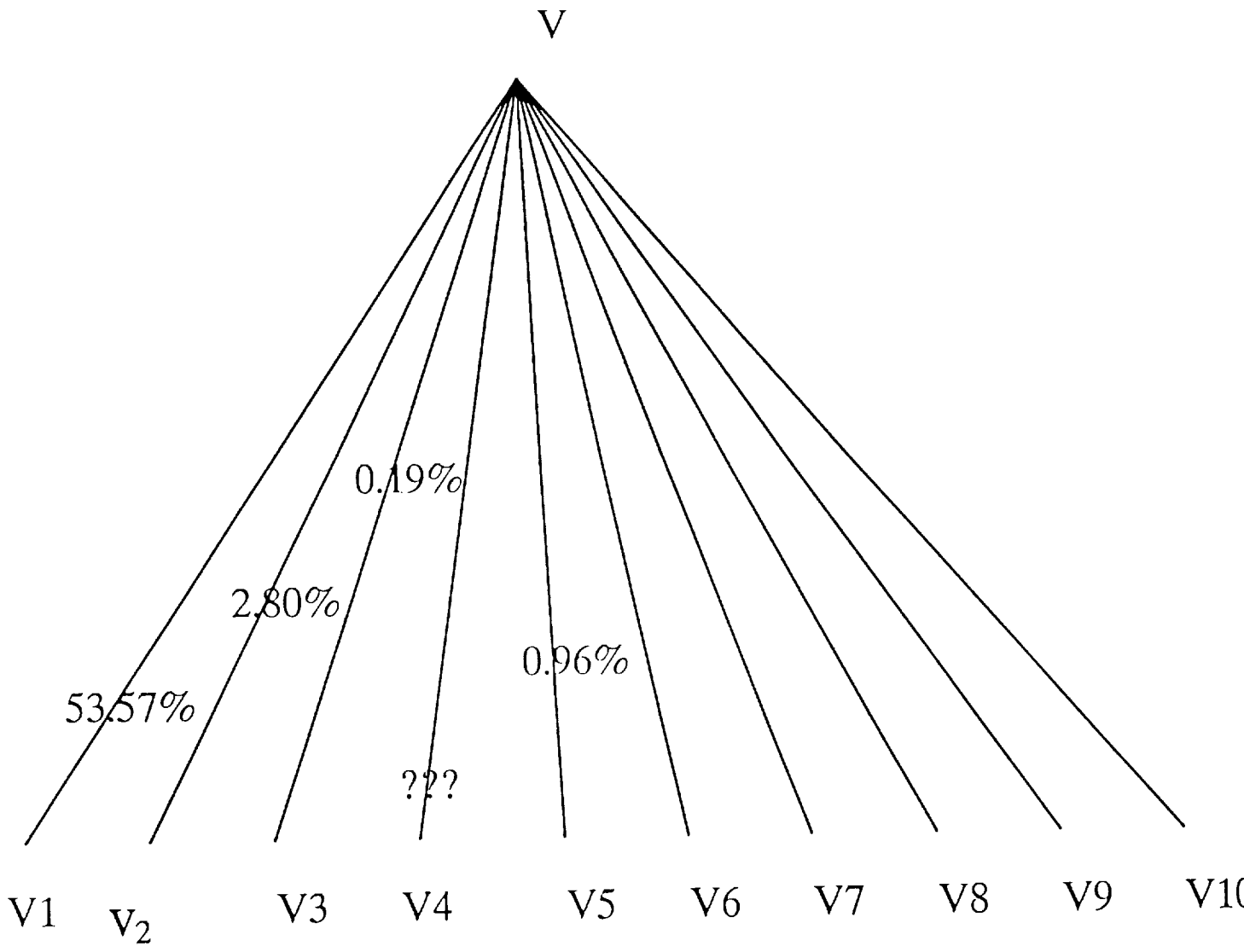
ΔRPDG9


45.729
V−1
97.254
V−2
99.797
V−3
98.618
V−4
98.984
V−5
99.847
V−6
99.967
V−7
99.998
V−8
99.998
V−9
99.99
V−1
54%
2.7%
0.2%
1.4%
0.15%
1.0%
0.03%
0.002%
0.002%
0.001%
V1..10
99.999%


V
ΔRPDG
DEPENDENCY
1.46%
0.553%
0.378%
0.064%
0.064%
51.9%
0.165%
0.539%
0.355%
0.064%
V5

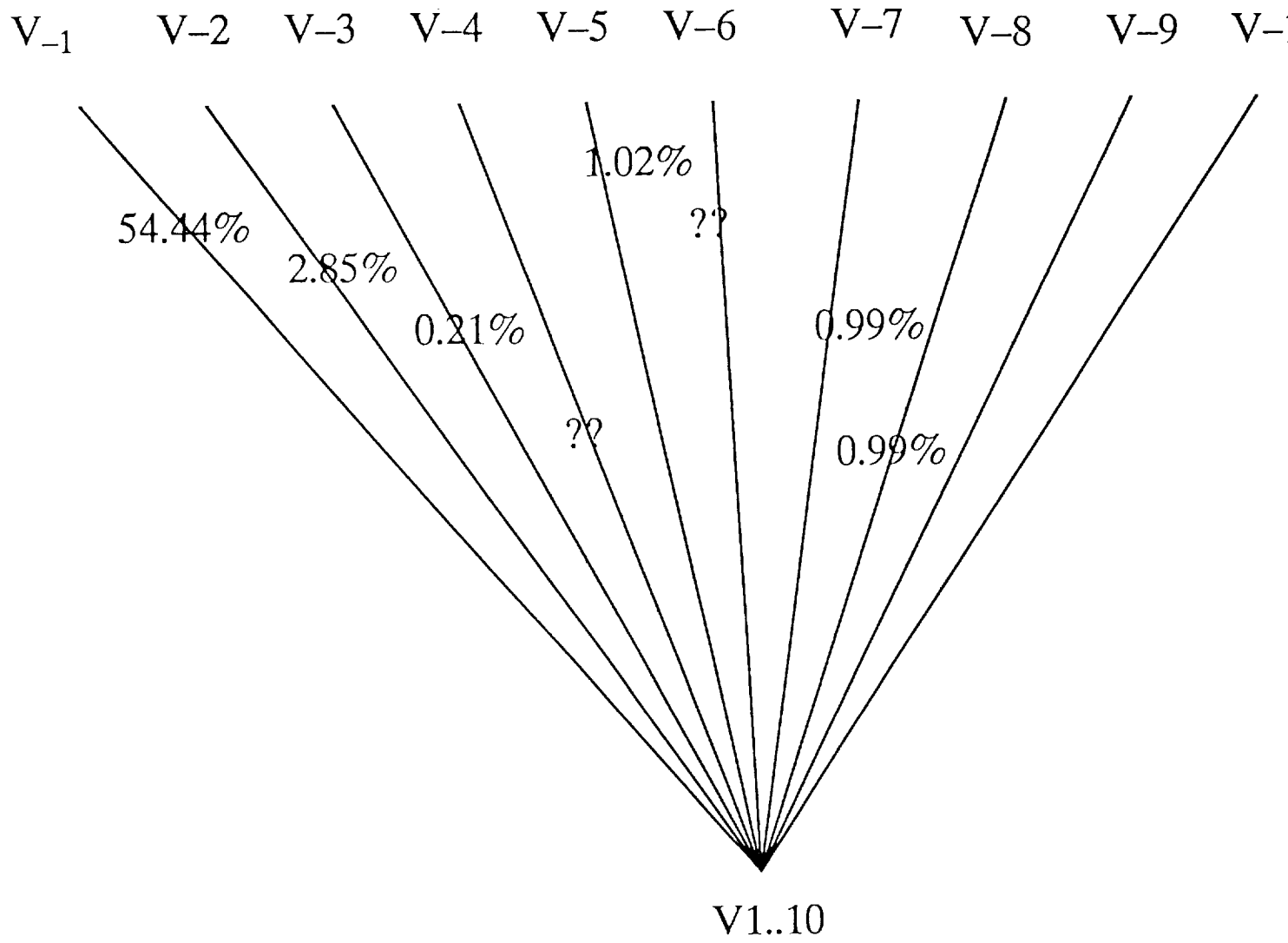

V-5
1.0%
0.03%
0.002%
0.2%
54.0%
2.7%
1.4%
0.15%
0.002%
0.001%
V1...10

ΔPDG1



ΔPDG9

ΔSPDG



FUTURE

1. ANALYZE BUG DEPENDENCY
   FILL IN ΔRPDG

2. NEW MODELS
   FILL IN ROWS 8 AND 9 ΔSRPDG
   ANALYZE

3. REPEAT FOR LIC 3.C

4. CONTINUE WITH GCS DATA