# Using Information Theory to Guide Fault Localisation

Shin Yoo (joint work with Mark Harman & David Clark)
CREST, UCL

FLINT: Fault Localisation using Information Theory
Shin Yoo, Mark Harman and David Clark
RN/11/09, Department of Computer Science, University College London, 2011

# Outline

- Shannon's Entropy
- How we make our (short?) prediction
- Empirical results

# What is entropy?

- Entropy = amount of uncertainty regarding a random variable
- Information = change in entropy (i.e. more knowledge is less uncertainty)

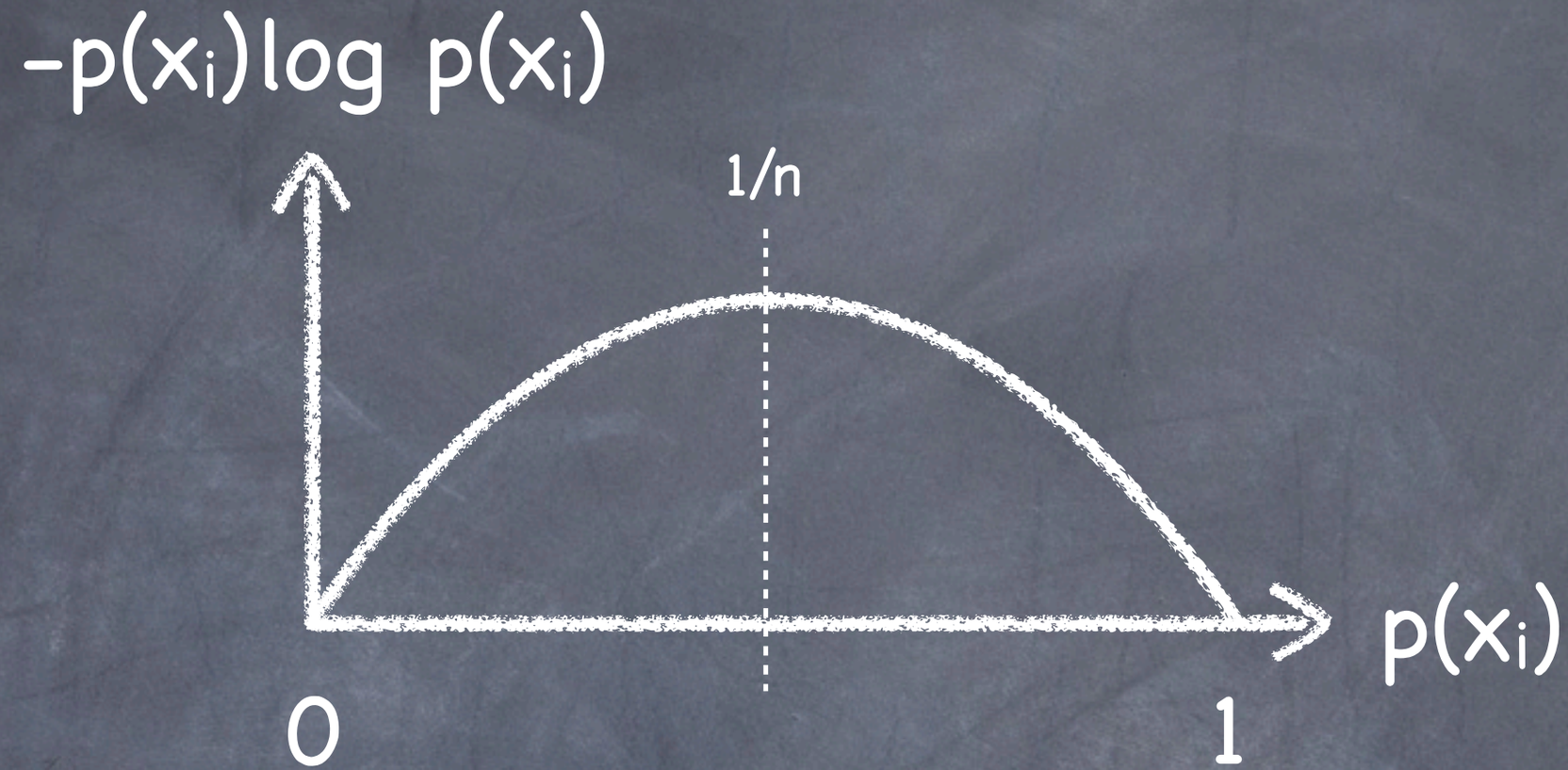
# What is entropy?

- Let X be one of $\{x_1, x_2, \ldots, x_n\}$
- If X is very likely to be $x_4$, i.e. $P(X=x_4) \approx 1$, there is little uncertainty
- Similarly, if X is very likely not to be $x_3$, i.e. $P(X=x_3) \approx 0$, there is little uncertainty
- If X can be any of $\{x_1, x_2, \ldots, x_n\}$, there is maximum uncertainty

# Mathematical Properties

- Continuity: so that a small change in probability results in a small change in entropy.
- Monotonicity: so that if all n cases are equally likely, H monotonically increases as n increases.
- Additivity: so that if a choice can be broken down to two successive choice, the original H can be expressed in a weighted sum.

A mathematical theory of communication, Shannon, 1948




$$H(X) = -\sum_{i=1}^{n} p(x_i) \cdot \log p(x_i)$$

To reduce entropy of X is to drive $p(x_i)$ to either 0 or 1 for each $x_i$. The amount of reduction is our information gain.

# Test-based Fault Localisation

- Given results of tests which include failing ones, how can we know where the faulty statement(s) lies in the program?

# FLINT: Fault Localisation using Information Theory

# Probabilistic Model of Fault Locality

- Program with m statements, $S=\{s_0, s_1, \ldots, s_{m-1}\}$
- Test suite with n tests, $T= \{t_0, t_1, \ldots, t_{n-1}\}$
- S contains a single fault
- Random variable X represents the locality

# Probabilistic Model of Fault Locality

- At the beginning of fault localisation:
  - P(X) = 1 / m : we suspect everything equally
  - H(X) = log(m) (the maximum)

# Probabilistic Model of Fault Locality

- At the end of fault localisation, "ideally":
  - $P(X=s_j) = 1$
  - $P(X \in S - \{s_j\}) = 0$
  - $H(X) = 0$ (i.e. no uncertainty)

# A quantitative view

- Fault localisation is all about making H(X) zero, or as little as possible
- H(X) measures your progress
- We can measure how much each test contributes to localisation, **provided that we build a probability distribution model of locality around tests**

# Localisation Metrics

- Also called "suspiciousness"
- Relative measure of how likely each statement is to contain the fault
- Often calculated from the execution traces of tests
- Tarantula, Ochiai, Jaccard, etc

# Tarantula metric

$$\text{Tarantula metric } \tau(s) = \frac{\frac{fail(s)}{totalfail}}{\frac{pass(s)}{totalpass} + \frac{fail(s)}{totalfail}}$$

- pass(s): # of passing tests that cover s
- fail(s): # of failing tests that cover s
- 1 if test fails whenever s is covered; 0 if test passes whenever s is covered

# Probability Distribution from Tarantula

$$\mathbf{P}_{T_i}(B(s_j)) = \frac{\tau(s_j|T_i)}{\sum_{j=1}^{m} \tau(s_j|T_i)}$$

- After executing up to test i, we take the normalised suspiciousness as the probability of locality

# Entropy from Tarantula

$$\mathbf{H}_{T_i}(S) = -\sum_{j=1}^{m} \mathbf{P}_{T_i}(B(s_j)) \cdot \log \mathbf{P}_{T_i}(B(s_j))$$

- Entropy of locality after executing up to $t_i$
- Suppose $t_i$ failed and we want to locate the fault: which test should we execute first?

# FLP

- Fault Localisation Prioritisation: prioritise tests according to the amount of information they reveal

:-)

"But how do you know how much information will be revealed BEFORE executing a test?"

:-(

# Predictive Modelling of Suspiciousness

$$\mathbf{P}_{T_{i+1}}(B(s_j)) = \mathbf{P}_{T_{i+1}}(B(s_j)|F(t_{i+1})) \cdot \alpha + \mathbf{P}_{T_{i+1}}(B(s_j)|\neg F(t_{i+1})) \cdot (1-\alpha)$$

$$\alpha = \mathbf{P}_{T_{i+1}}(F(t_{i+1})) \approx \frac{TF_i}{TP_i + TF_i}$$

- For each statement $s_j$, it either contains fault or not
- For each unexecuted test $t_i$, it either passes or fail
- $P_{Ti+1}(B(s_j)|F(t_{i+1}))$ and $P_{Ti+1}(B(s_j)|\sim F(t_{i+1}))$ are approximated with Tarantula

# Predictive Modelling of Suspiciousness

- Once we can predict the probability of fault locality for each test, we can also predict the entropy
- Once we predict the entropy, we can predict which test will yield the largest information gain

# Total Information Retain

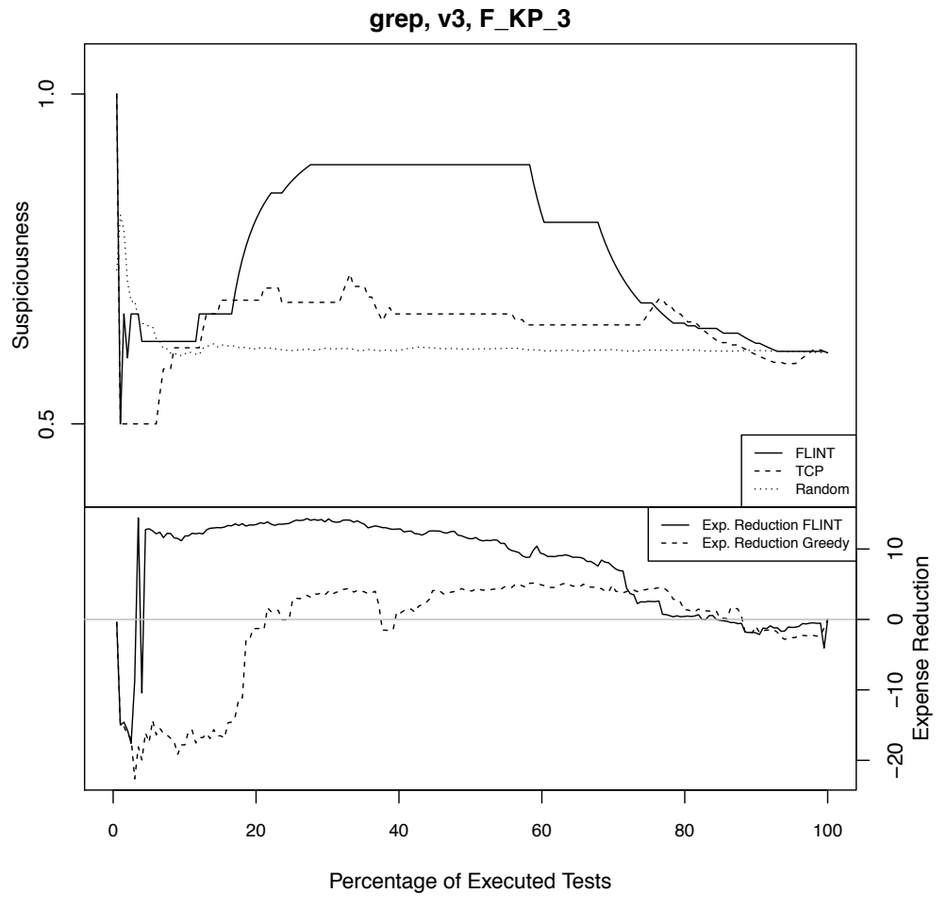
- Yet the total information yielded by a test suite retain (that is, at the end of testing, the information we get out of the activity remains the same, whichever ordering of tests we take).
- So why bother?
- It's the ordering that matters!

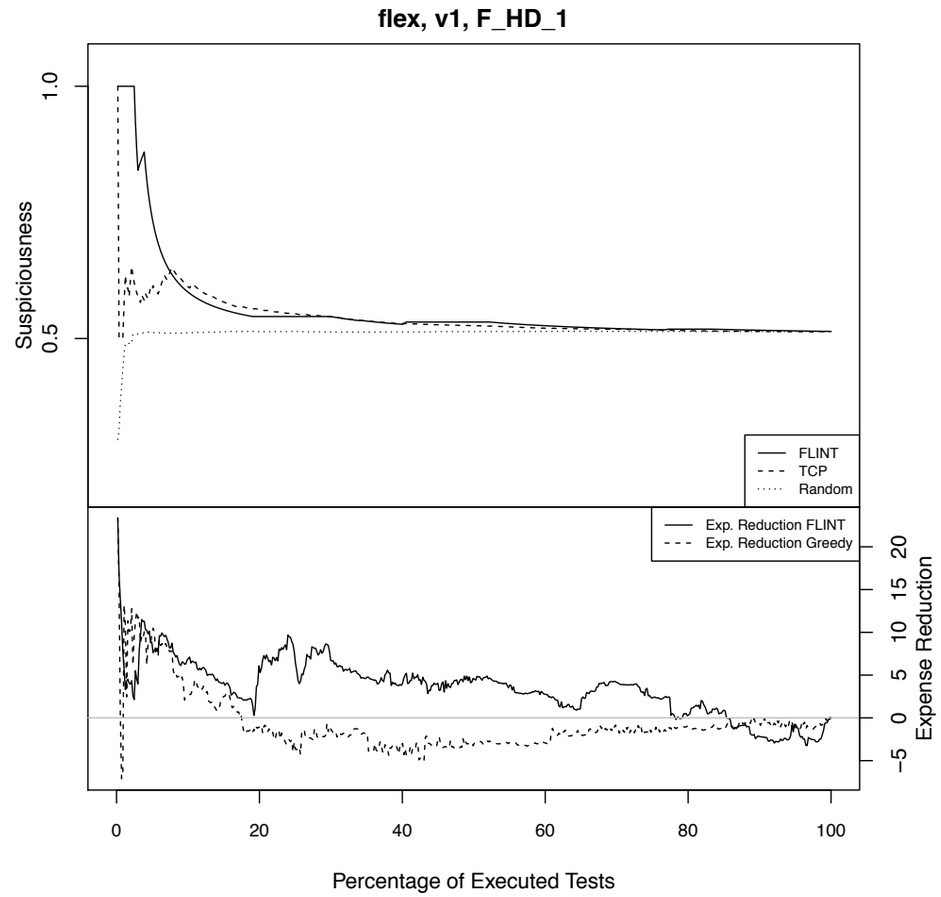
# Empirical Study

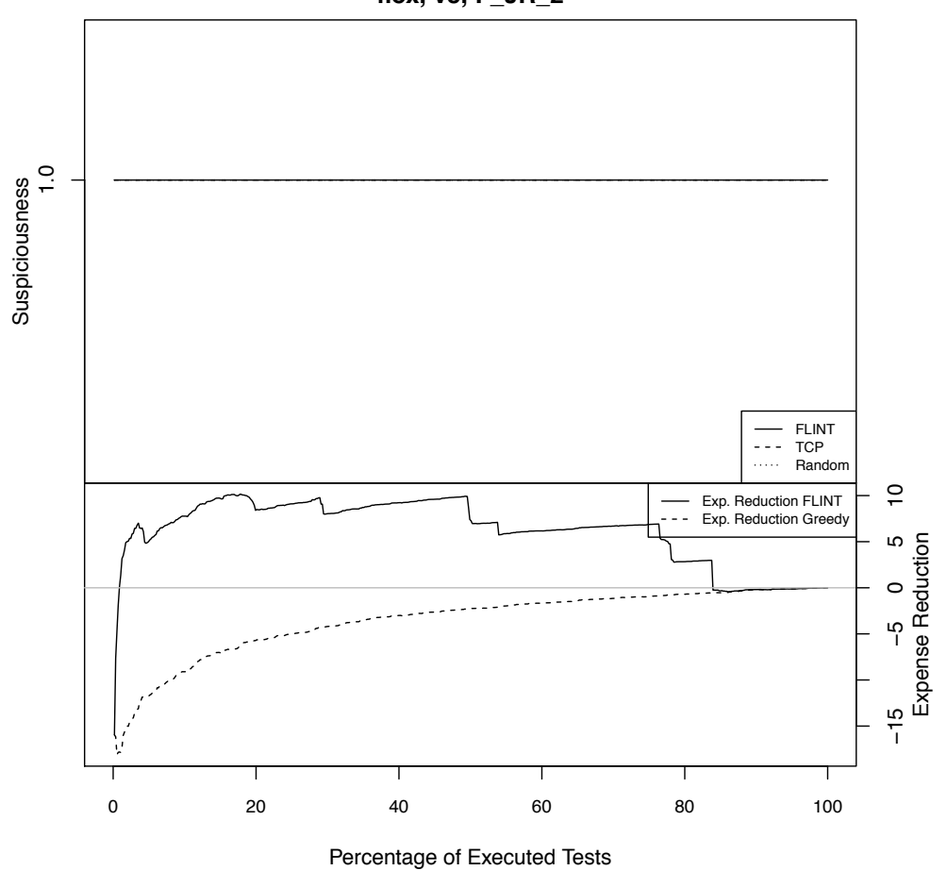
- 92 faults from 5 consecutive versions of flex, grep, gzip and sed
- Compared to random and coverage-based prioritisation (normal TCP, not FLP)

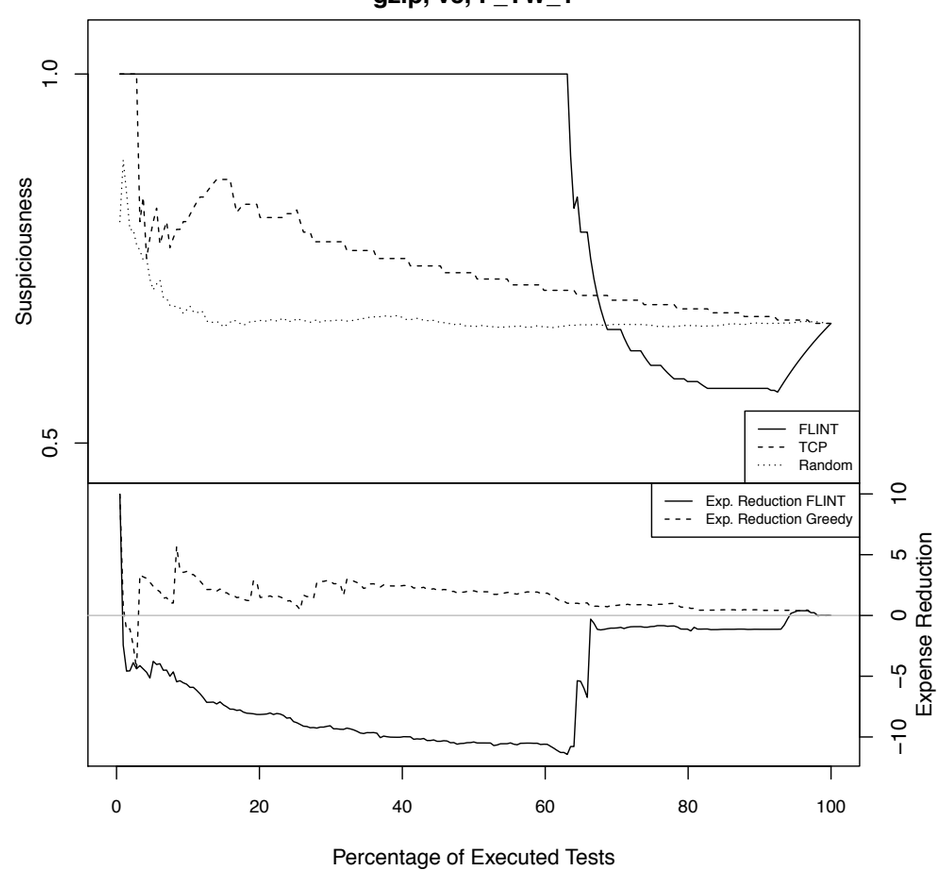
# Effectiveness Measure

- Expense = (rank of faulty statement) / m * 100
- Measures how many statements the tester has to consider, following the suspiciousness ranking, until encountering the faulty one


grep, v3, F_KP_3
Suspiciousness
1.0
0.5
FLINT
TCP
Random
Exp. Reduction FLINT
Exp. Reduction Greedy
Expense Reduction
10
0
−10
−20
0
20
40
60
80
100
Percentage of Executed Tests


flex, v1, F_HD_1
Suspiciousness
1.0
0.5
FLINT
TCP
Random
Exp. Reduction FLINT
Exp. Reduction Greedy
Expense Reduction
20
15
10
5
0
−5
0
20
40
60
80
100
Percentage of Executed Tests


flex, v5, F_JR_2
Suspiciousness
1.0
FLINT
TCP
Random
Exp. Reduction FLINT
Exp. Reduction Greedy
Expense Reduction
10
5
0
−5
−15
0
20
40
60
80
100
Percentage of Executed Tests


gzip, v5, F_TW_1
Suspiciousness
1.0
0.5
FLINT
TCP
Random
Exp. Reduction FLINT
Exp. Reduction Greedy
Expense Reduction
10
5
0
−5
−10
0
20
40
60
80
100
Percentage of Executed Tests

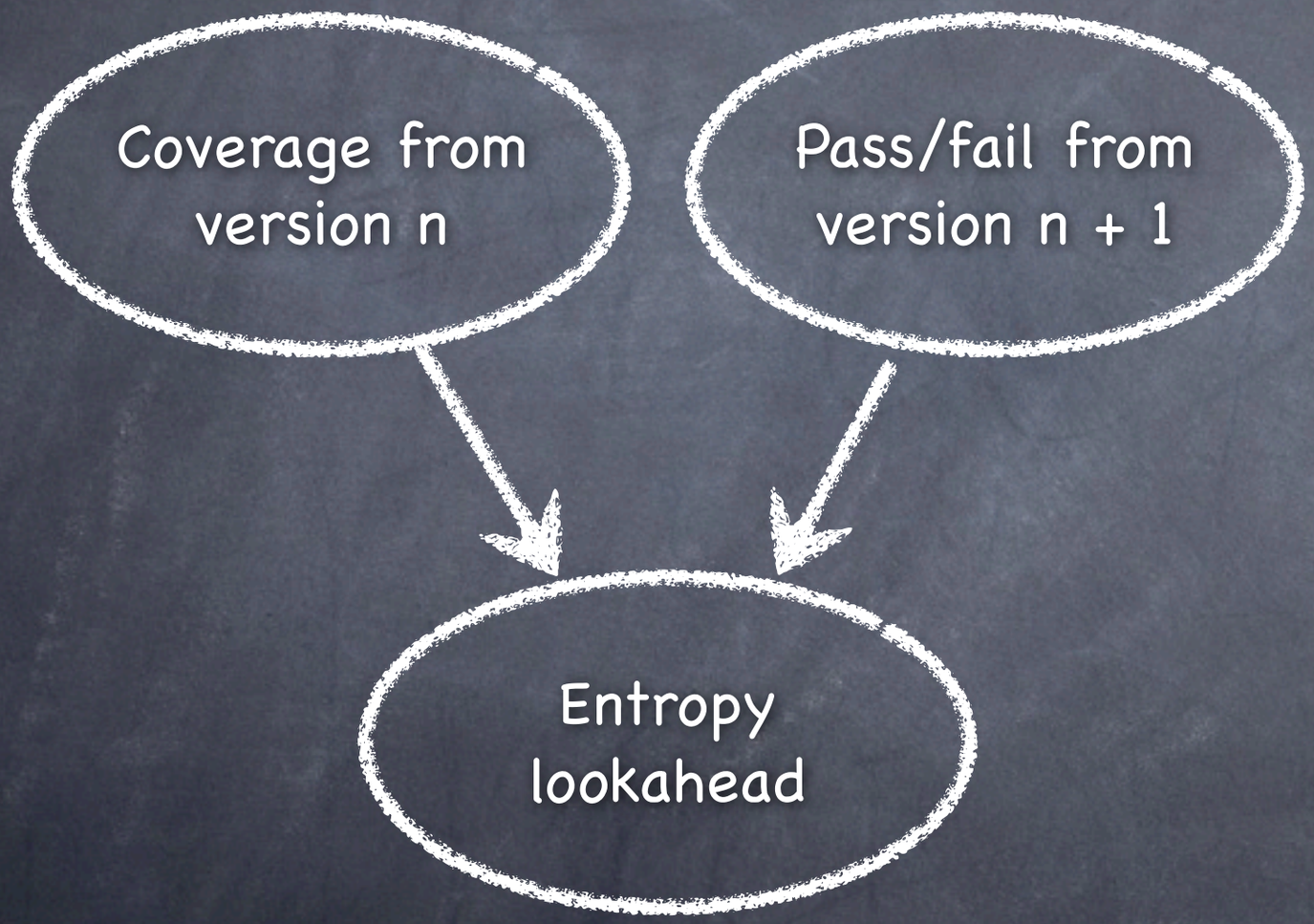
# Statistical Comparisons

| | PS | PN | EQ | NN | NS |
|---|---|---|---|---|---|
| $E_T < E_R$ | 70.65% | 1.09% | 0% | 0% | 28.26% |
| $E_F < E_R$ | 73.91% | 2.17% | 0% | 0% | 23.91% |
| $E_F < E_T$ | 46.74% | 2.17% | 10.87% | 6.52% | 33.70% |

# When coverage is unknown

- Remember we said "$P_{Ti+1}(B(s_j)|F(t_{i+1}))$ and $P_{Ti+1}(B(s_j)|\sim F(t_{i+1}))$ are approximated with Tarantula"
- That is only possible if we know which statement ti +1 covers
- Which is not known when you run your test for a new version!

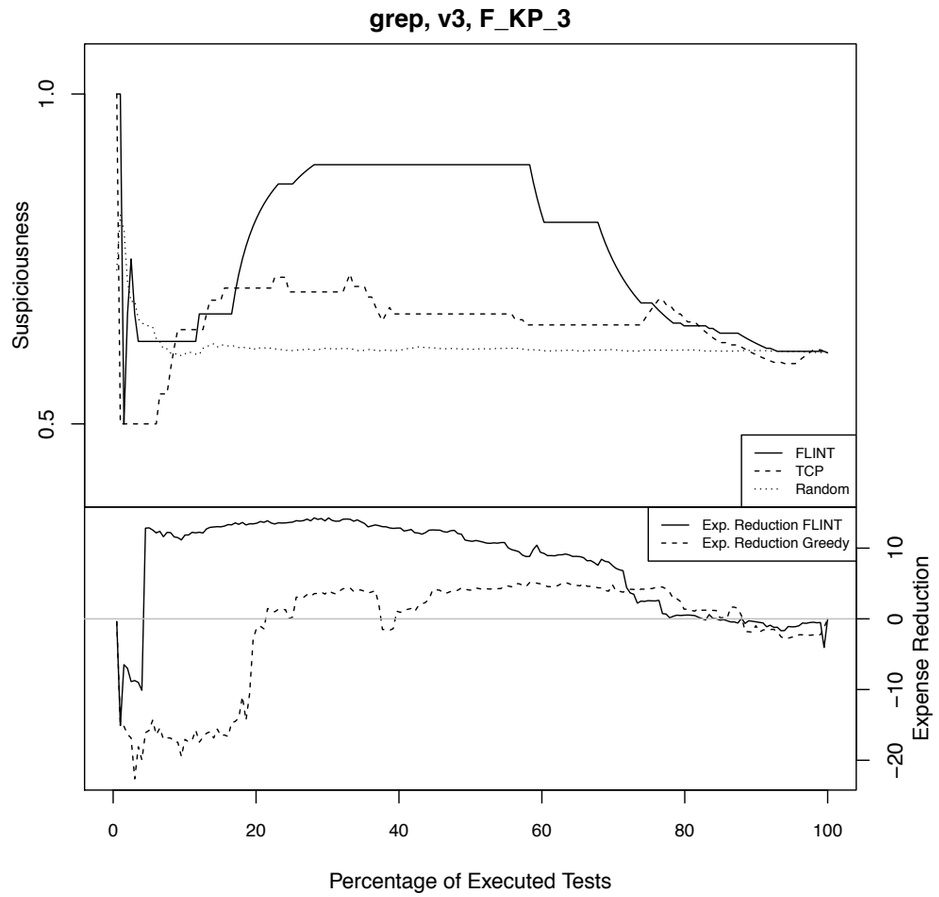
# When coverage is unknown

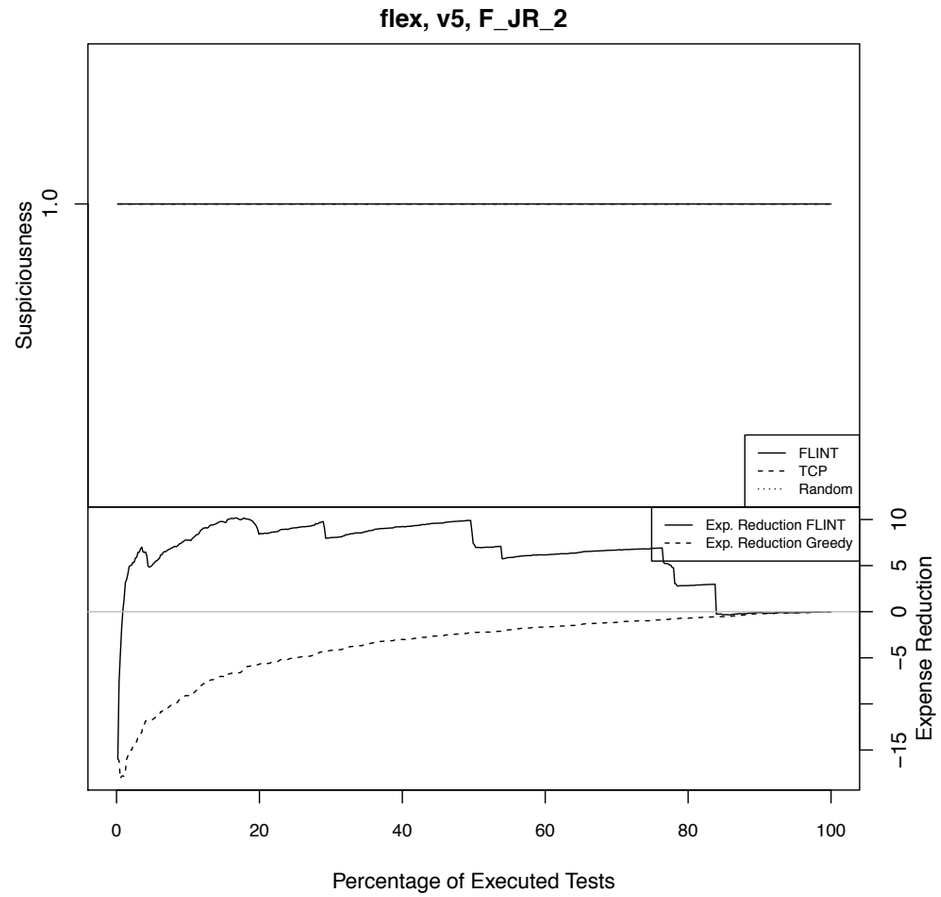


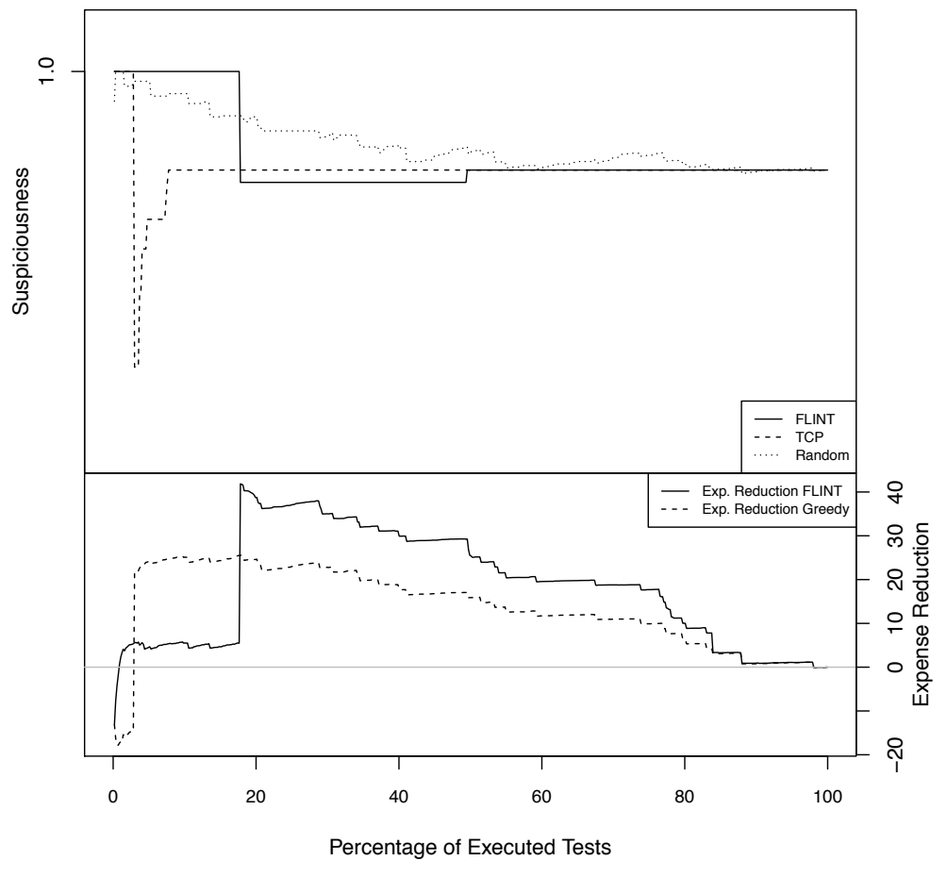

- We use coverage from previous version, i.e. localise the fault w.r.t. the previous version
- We only take actual pass/fail result from current version

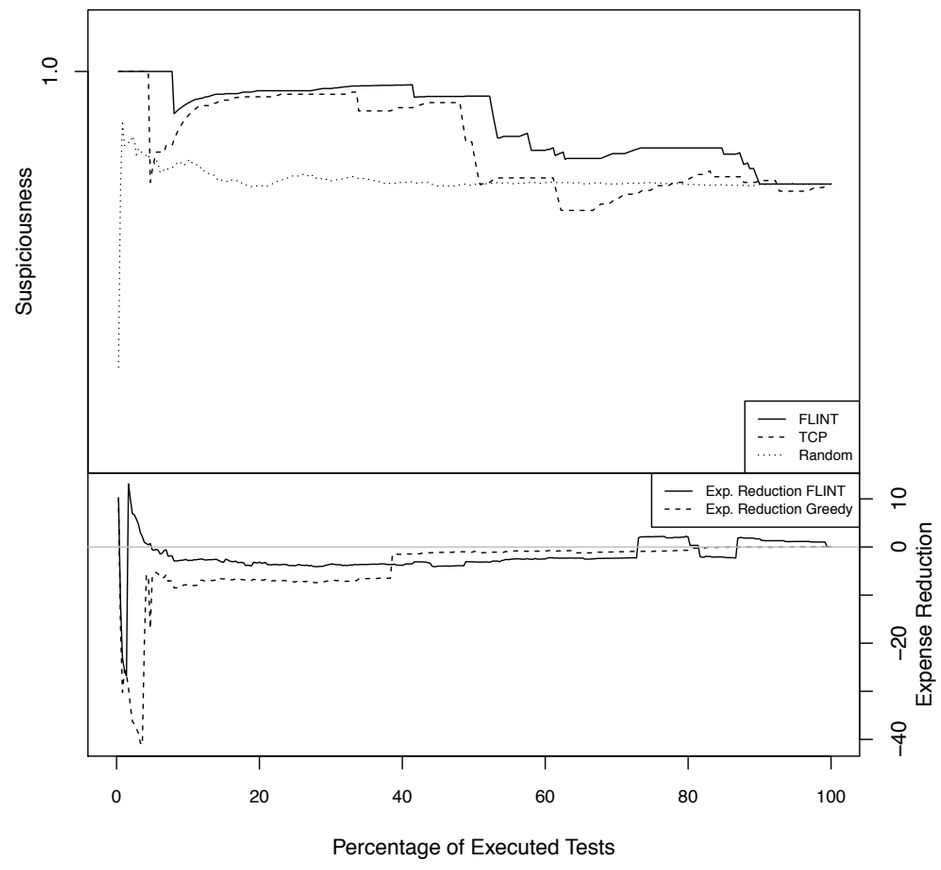
# "Nonsense!"

- No, it is possible because our approach only guides the probability distribution: it does not concern any specific statement, how many statements there are, etc


grep, v3, F_KP_3
Suspiciousness
1.0
0.5
FLINT
TCP
Random
Exp. Reduction FLINT
Exp. Reduction Greedy
10
0
−10
−20
Expense Reduction
0
20
40
60
80
100
Percentage of Executed Tests


flex, v5, F_JR_2
Suspiciousness
1.0
FLINT
TCP
Random
Exp. Reduction FLINT
Exp. Reduction Greedy
10
5
0
−5
−15
Expense Reduction
0
20
40
60
80
100
Percentage of Executed Tests


flex, v5, F_AA_4
Suspiciousness
1.0
FLINT
TCP
Random
Exp. Reduction FLINT
Exp. Reduction Greedy
40
30
20
10
0
−20
Expense Reduction
0
20
40
60
80
100
Percentage of Executed Tests


sed, v2, F_AG_19
Suspiciousness
1.0
FLINT
TCP
Random
Exp. Reduction FLINT
Exp. Reduction Greedy
10
0
−20
−40
Expense Reduction
0
20
40
60
80
100
Percentage of Executed Tests

# Use Case

- You've already run all tests and detected a failure, you want to check results to locate the fault. Which "checking" order do you follow?
  - Use FLINT with actual coverage data
- You are in the middle of testing, a failure has been detected, you want to prioritise the remaining tests to locate the fault asap. Which order do you follow?
  - Use FLINT with previous coverage data

# "What about multiple faults?"

- Again, we benefit from the generic nature of entropy: it never concerns any specific faults
- It is not unrealistic to assume that the tester can distinguish different faults: filter pass/fail results accordingly into FLINT

# "But Tarantula is weak"

- FLINT only requires a probability distribution: we evaluated it with Tarantula because it is intuitive and easy to calculate
- More sophisticated fault localisation metric will only improve FLINT
- Many opportunities for short-term prediction/speculation

# Conclusion

- Shannon's entropy is not only beautiful but actually useful for fault localisation
- It is very universal and powerful at the same time and we encourage you to consider it to frame your own research agenda