Duals in spectral bug localization

Lee Naish
Computing and Information Systems
University of Melbourne

Hua Jie (Jason) Lee
Dolby Laboratories

These slides are on the web:

http://www.cs.mu.oz.au/˜lee/papers/dual/

Contribution

Primarily theoretical:

Literature on set similarity measures discusses various forms of symmetry and other properties

We apply these ideas to obtain new similarity measures for fault localization and gain a deeper understanding of the problem

Outline

Spectra-Based Fault Localization

Execute the program multiple times using a test suite where we can tell if each result is correct or not, gathering data about each execution

For each statement/. . . , estimate how likely it is to be buggy based on the data gathered

Rank the statements accordingly, then check the code manually, starting with the highest ranked statement until the bug is found (or we give up)

Cost measure: given a ranking of S statements, the *rank cost* is

$$\frac{GT + EQ/2}{S}$$

where GT is the number of correct statements ranked strictly higher than all bugs and EQ is the number of correct statements ranked equal to the highest ranked bug

Statement spectra

Collect data on which statements are executed for each test

We count

- The total number of failed tests, F,
- The total number of passed tests, P,

and for each statement S_i, the number of

- failed tests in which it was executed, a^i_{ef},
- passed tests in which it was executed, a^i_{ep}.
- failed tests in which it was not executed, a^i_{nf}, and
- passed tests in which it was not executed, a^i_{np}.

(the last two could be implicit)

Statement spectra example

The raw data is a binary matrix (1 means the statement was executed in the test) and a binary vector (1 means the test failed)

We compute F, P and the a_{ef} and a_{ep} for each statement, eg

	T_1	T_2	T_3	T_4	T_5	a_{ef}	a_{ep}
S_1	1	0	0	1	0	1	1
S_2	1	1	0	1	0	2	1
S_3	1	1	1	0	1	2	2
				⋮			
Res.	1	1	0	0	0	$F = 2$	$P = 3$

To rank statements, measure "similarity" of matrix rows and result vector

Some similarity metrics used for ranking

Name	Formula	Name	Formula
Tarantula	$\frac{\frac{a_{ef}}{a_{ef}+a_{nf}}}{\frac{a_{ef}}{a_{ef}+a_{nf}}+\frac{a_{ep}}{a_{ep}+a_{np}}}$	Jaccard	$\frac{a_{ef}}{a_{ef}+a_{nf}+a_{ep}}$
Ochiai	$\frac{a_{ef}}{\sqrt{(a_{ef}+a_{nf})(a_{ef}+a_{ep})}}$	Hamming	$a_{ef} + a_{np}$
O^p	$a_{ef} - \frac{a_{ep}}{a_{ep}+a_{np}+1}$	Ample2	$\frac{a_{ef}}{a_{ef}+a_{nf}} - \frac{a_{ep}}{a_{ep}+a_{np}}$

O^p was proven optimal for a model single-bug program

A metric M is *single bug optimal* if, when $a_{ef} = F$,

1. the value returned is always greater than any value returned when $a_{ef} < F$, and
2. M is strictly decreasing in a_{ep}

Two properties metrics should have

1) Well-definedness for all possible a_{ij} values

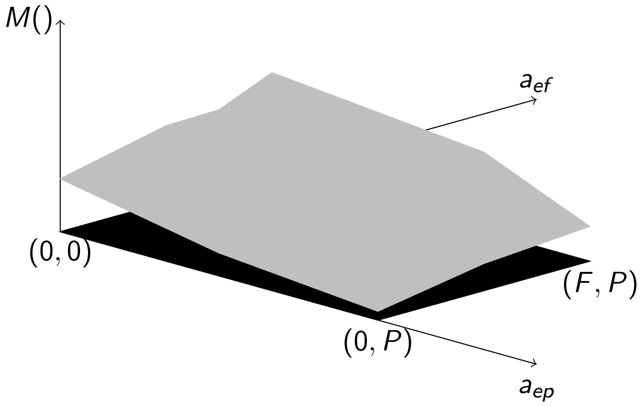
2) Monotonicity: strictly increasing in a_{ef} and decreasing in a_{ep}, for a fixed F and P

By adding a tiny constant ϵ in various places we can tweak the metrics to ensure well-definedness and monotonicity (Tarantula is equivalent to $\frac{a_{ef}}{a_{ep}}$)

Name	Formula	Name	Formula
T	$\frac{a_{ef}+\epsilon}{a_{ep}+\epsilon}$	J	$\frac{a_{ef}+\epsilon}{a_{ef}+a_{nf}+a_{ep}}$
C	$\frac{a_{ef}+\epsilon}{\sqrt{(a_{ef}+a_{nf})(a_{ef}+a_{ep}+\epsilon)}}$	H	$a_{ef}+a_{np}$
N	$a_{ef}-\epsilon a_{ep}$	A	$\frac{a_{ef}}{a_{ef}+a_{nf}}-\frac{a_{ep}}{a_{ep}+a_{np}}$

A graphical view

Given fixed F and P, a similarity measure can be viewed as a surface over the domain of a_{ef} and a_{ep} values



More properties metrics may have: symmetries

Suppose $\mathcal{E}$ is the set of tests which execute a given statement and $\mathcal{F}$ is the set of tests which fail

Saying $\mathcal{E}$ is similar to $\mathcal{F}$ is like saying the complement of $\mathcal{E}$ is dis-similar to $\mathcal{F}$, or $\mathcal{E}$ is dis-similar to the complement of $\mathcal{F}$, or the complement of $\mathcal{E}$ is similar to the complement of $\mathcal{F}$

We define the *E-dual*, *F-dual* and *EF-dual* of a metric M:

$$\mathcal{D}^E(M)(a_{ef}, a_{ep}, a_{nf}, a_{np}) = -M(a_{nf}, a_{np}, a_{ef}, a_{ep})$$

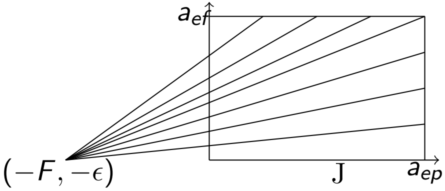
$$\mathcal{D}^F(M)(a_{ef}, a_{ep}, a_{nf}, a_{np}) = -M(a_{ep}, a_{ef}, a_{np}, a_{nf})$$

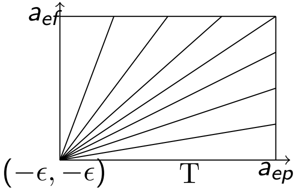
$$\mathcal{D}^{EF}(M)(a_{ef}, a_{ep}, a_{nf}, a_{np}) = M(a_{np}, a_{nf}, a_{ep}, a_{ef})$$

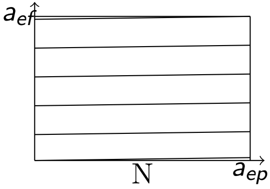
Metrics are equivalent to zero, one or all three of their duals

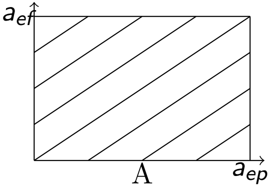
Symmetries and contour plots

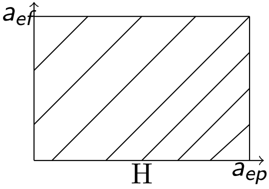
$\mathcal{D}^E(\mathrm{N})$ and N are equivalent, $\mathcal{D}^E(\mathrm{A})$, $\mathcal{D}^F(\mathrm{A})$, $\mathcal{D}^{EF}(\mathrm{A})$, and A are equivalent, $\mathcal{D}^E(\mathrm{H})$, $\mathcal{D}^F(\mathrm{H})$, $\mathcal{D}^{EF}(\mathrm{H})$, and H are equivalent, and $\mathcal{D}^F(\mathrm{T})$ and T are equivalent











Optimality for single bugs and deterministic bugs

$\mathcal{D}^E(\mathrm{J})$ and $\mathcal{D}^E(\mathrm{C})$ are both single bug optimal

$\mathcal{D}^E(\mathrm{T})$ is single bug optimal if statements which are executed in all tests are ignored

For a single bug, $a_{ef} = F$ (and $a_{nf} = 0$)

Deterministic bugs cause failure whenever they are executed: $a_{ep} = 0$ (and $a_{np} = P$) — the *EF-dual* of the single bug case

A metric M is *deterministic bug optimal* if, when $a_{ep} = 0$, the value returned is always greater than any value returned when $a_{ep} > 0$, and M is strictly increasing in a_{ef}

$\mathcal{D}^F(\mathrm{N})$, $\mathcal{D}^{EF}(\mathrm{N})$, $\mathcal{D}^F(\mathrm{J})$, and $\mathcal{D}^F(\mathrm{C})$ are deterministic bug optimal, as is T if statements which are executed in no tests are ignored

Empirical validation of theory (Siemens + Unix)

Metric	STS	Unix	Combined
N	14.78	19.37	16.87
$\mathcal{D}^E$(C)	14.78	19.37	16.87
$\mathcal{D}^E$(J)	14.78	19.37	16.87
$\mathcal{D}^E$(*Jaccard*)	27.46	30.90	29.02
$\mathcal{D}^E$(T)	16.85	21.52	18.98
C	19.26	22.28	20.63
J	22.57	22.75	22.65
Jaccard	22.57	22.75	22.65
A	21.30	25.23	23.09
T	23.22	28.17	25.47
Tarantula	23.22	31.49	26.98
H	43.57	29.37	37.10
$\mathcal{D}^{EF}$(C)	43.87	32.51	38.70
$\mathcal{D}^{EF}$(J)	44.11	32.93	39.02
$\mathcal{D}^{EF}$(*Jaccard*)	44.12	32.93	39.02
$\mathcal{D}^F$(C)	44.04	34.92	39.88
$\mathcal{D}^F$(J)	44.35	35.23	40.20

Empirical validation of theory (model results)

	Single bug		Det. bug	
Num. Tests	20	50	20	50
N	3.01	1.47	2.85	2.79
$\mathcal{D}^E$(J)	3.01	1.47	2.64	2.33
$\mathcal{D}^E$(*Jaccard*)	5.30	2.78	2.76	2.35
$\mathcal{D}^E$(T)	3.07	1.48	1.49	1.06
J	4.28	3.27	1.77	1.43
Jaccard	4.22	3.25	1.77	1.43
A	4.33	3.04	0.87	0.49
T	6.14	4.92	0.76	0.35
Tarantula	6.12	4.90	3.50	1.85
H	9.98	8.85	1.47	1.16
$\mathcal{D}^{EF}$(J)	13.69	12.33	0.94	0.61
$\mathcal{D}^{EF}$(*Jaccard*)	13.76	12.33	0.93	0.61
$\mathcal{D}^F$(J)	14.93	13.71	0.76	0.35
$\mathcal{D}^F$(*Jaccard*)	15.07	13.72	3.54	1.85
$\mathcal{D}^F$(N)	17.77	17.43	0.76	0.35

How can we choose a good similarity metric?

Conjecture: In early stages of software development, metrics that have contours with relatively high gradients are preferable, whereas in late stages of software development, metrics that have contours with relatively low gradients are preferable

In early stages, multiple bugs are expected and deterministic bugs are relatively likely. In late stages we have (hopefully) eliminated deterministic bugs and are converging towards the single bug case

Conclusions

Instead of looking for a single “best” similarity metric for fault localization, we should choose different metrics based on the stage of software development

Sometimes theoretical nit picking can have significant practical benefits