A Novel Framework for Locating Software Faults Using Latent Divergences

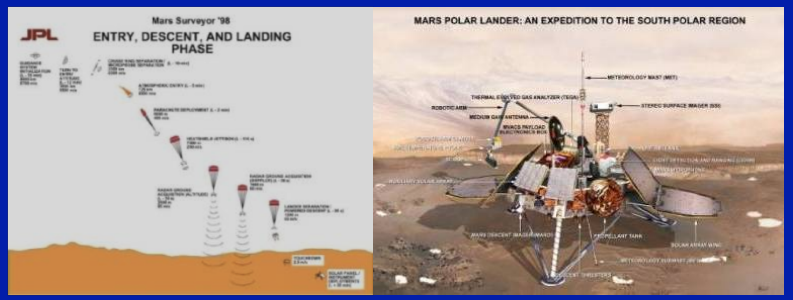
Shounak Roychowdhury Sarfraz Khurshid
University of Texas at Austin

ECML PKDD, Athens
6 September 2011

Software failures are expensive

The cost is particularly high for mission-critical systems

Photos: *ESA/CNES*



Photos: *JPL/NASA*

Photo: *navsource.org*

NIST 2002: Software flaws cost the US economy $60B/yr

- Better testing infrastructure could save $20B/yr

Testing and Debugging

Software testing: finding failures in execution of code

- Most commonly used technique for validating software quality
- Conceptually simple: get inputs; run program; check outputs
- In practice: expensive and often ineffective

Debugging: locating and removing faults in source code

- Required for removing bugs in code
  - Even if bugs found using static analysis, inspection etc.
- Conceptually simple: inspect failure; locate faults; fix them
- In practice: tedious and error-prone
  - Inspecting a small portion of a large program can be hard

Overview

Problem – given labeled (correct and erroneous) execution traces, assist with debugging

Insight – fault localization (SE) reduces to feature selection (ML)

- Fault localization – select a set of most relevant statements
  - Feature selection in machine learning provides a solution

Methodology

- Model code coverage data as a probabilistic data source
- Use probabilistic divergences, e.g., KL, α, *f*, and Bregman, to detect faulty lines of code
  - Latent divergence: product of divergences based on conditional probabilities
    - Inspired by power of a point with respect to a circle

Framework – a family of measures

- Experimental evaluation shows effectiveness

Outline

Overview

Example

Framework

Experiments

Conclusion

Example
faulty program*

		3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3
	`mid() {`						
	`int x,y,z,m;`						
1:	`read("Enter 3 numbers:",x,y,z);`	•	•	•	•	•	•
2:	`m = z;`	•	•	•	•	•	•
3:	`if (y<z)`	•	•	•	•	•	•
4:	`if (x<y)`	•	•			•	•
5:	`m = y;`		•				
6:	`else if (x<z)`	•				•	•
7:	`m = y; // fault (m = x)`	•					•
8:	`else`			•	•		
9:	`if (x>y)`			•	•		
10:	`m = y;`			•			
11:	`else if (x>z)`				•		
12:	`m = x;`						
13:	`print("Middle number is:", m);`	•	•	•	•	•	•
	`}`						
	Pass Status:	P	P	P	P	P	F

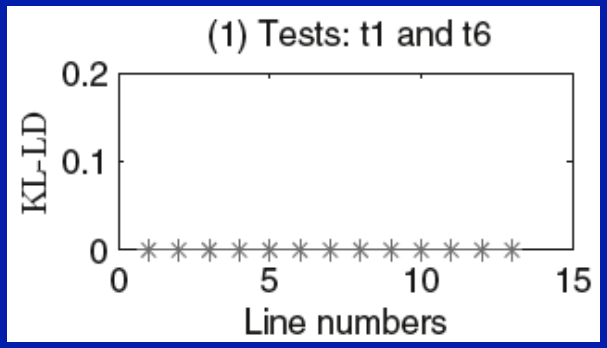
*Example C code from Jones and Harrold [ASE 2005]

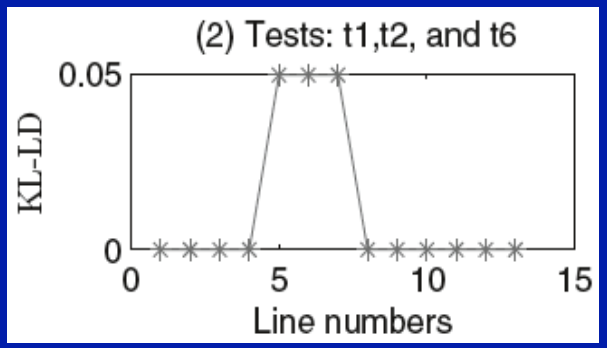
Example
fault localization using latent divergence

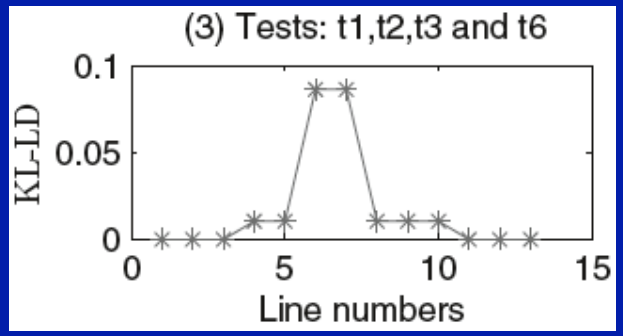
```
    mid() {
      int x,y,z,m;
1:    read("Enter 3 numbers:",x,y,z);
2:    m = z;
3:    if (y<z)
4:      if (x<y)
5:            m = y;
6:      else if (x<z)
7:            m = y; // fault (m = x)
8:    else
9:      if (x>y)
10:           m = y;
11:     else if (x>z)
12:           m = x;
13:   print("Middle number is:", m);
    }
```

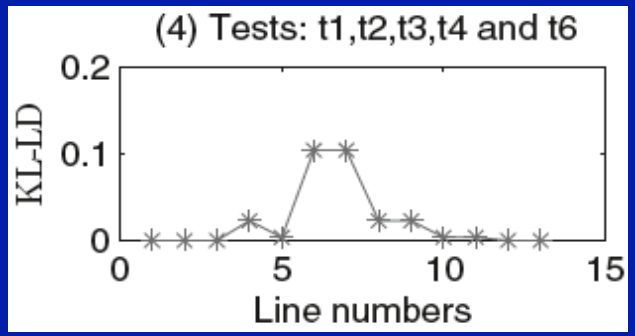
3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3
•	•	•	•	•	•
•	•	•	•	•	•
•	•	•	•	•	•
•	•			•	•
	•				
•				•	•
•					•
		•	•		
		•	•		
		•			
			•		
•	•	•	•	•	•
P	P	P	P	P	F

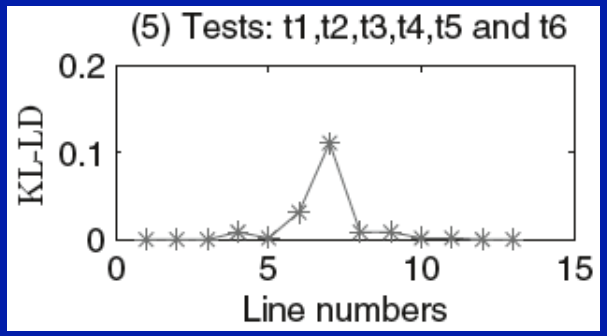
Pass Status:











Example
effectiveness of different techniques

		3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3
	`mid() {`						
	`int x,y,z,m;`						
1:	`read("Enter 3 numbers:",x,y,z);`	•	•	•	•	•	•
2:	`m = z;`	•	•	•	•	•	•
3:	`if (y<z)`	•	•	•	•	•	•
4:	`if (x<y)`	•	•			•	•
5:	`m = y;`		•				
6:	`else if (x<z)`	•				•	•
7:	`m = y; // fault (m = x)`	•					•
8:	`else`			•	•		
9:	`if (x>y)`			•	•		
10:	`m = y;`			•			
11:	`else if (x>z)`				•		
12:	`m = x;`						
13:	`print("Middle number is:", m);`	•	•	•	•	•	•
	`}`						
	Pass Status:	P	P	P	P	P	F

	1	2	3	4	5	6	7	8	9	10	11	12	13
Tarantula	0.50	0.50	0.50	0.625	0	0.71	0.83	0	0	0	0	0	0.5
Ochiai	0.7071	0.7071	0.7071	0.7906	0	0.8452	0.9129	0	0	0	0	0	0.7071
KL-$\mathcal{LD}$	0	0	0	0.0095	0.0037	0.0444	0.2110	0.0160	0.0160	0.0037	0.0037	0	0
JS-$\mathcal{LD}$	0	0	0	0.0366	0.0064	0.1312	0.4607	0.0366	0.0366	0.0064	0.0064	0	0
R(2)-$\mathcal{LD}$	0	0	0	0.0089	0.0015	0.0332	0.1072	0.0089	0.0089	0.0015	0.0015	0	0
R(0.5)-$\mathcal{LD}$	0	0	0	0.0019	0.0003	0.0069	0.0244	0.0019	0.0019	0.0003	0.0003	0	0
IS-$\mathcal{LD}$	0	0	0	0.8730	0.1617	2.8838	8.8456	0.8730	0.8730	0.1617	0.1617	0	0

Outline

Overview

Example

Framework

Experiments

Conclusion

Basis – Probabilistic Divergence ($p \parallel q$)

Measures distance between two probability mass functions

Several classes exist

- Kullback–Leibler divergence $$D_{KL}(p \parallel q) = \sum_{x \in \chi} p(x) \log\left(\frac{p(x)}{q(x)}\right)$$
- α-divergence $$D_{\alpha}(p \parallel q) = \frac{1}{1-\alpha} \log \sum_{x \in \chi} \left(\frac{p(x)^{\alpha}}{q(x)^{\alpha-1}}\right)$$
- *f*-divergence $$D_{f}(p \parallel q) = \sum_{x \in \chi} q(x) f\left(\frac{p(x)}{q(x)}\right)$$
- Bregman divergence
  - Itakura–Saito distance $$D_{IS}(p \parallel q) = \sum_{x \in \chi} \left(\frac{p(x)}{q(x)} - \log\left(\frac{p(x)}{q(x)}\right) - 1\right)$$

Key Definition – Latent Divergence

$r \in \{0,1\}, x \in \{0,1\}$

$p_R = \Pr(R = r)$

$p(x,r) = \Pr(X_l = x, R = r)$

$p_{\{R|X_l=1\}}(x) = \Pr(\{R \mid X_l = 1\} = x)$

$p_{\{R|X_l=0\}}(x) = \Pr(\{R \mid X_l = 0\} = x)$

	mid() { int x,y,z,m;	3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3
1:	read("Enter 3 numbers:",x,y,z);	•	•	•	•	•	•
2:	m = z;	•	•	•	•	•	•
3:	if (y<z)	•	•	•	•	•	•
4:	if (x<y)	•	•			•	•
5:	m = y;		•				
6:	else if (x<z)	•				•	•
7:	m = y; // fault (m = x)	•					•
8:	else			•	•		
9:	if (x>y)			•	•		
10:	m = y;			•			
11:	else if (x>z)				•		
12:	m = x;						
13:	print("Middle number is:", m);	•	•	•	•	•	•
	} Pass Status:	P	P	P	P	P	F

$$LD(X_l : R) = D(p_{\{R|X_l=1\}}(x) \,\|\, p_R) \times D(p_{\{R|X_l=0\}}(x) \,\|\, p_R)$$

Latent divergence is convex

Family of Latent Divergences

$$LD(X_l : R) = D(p_{\{R|X_l=1\}}(x) \,||\, p_R) \times D(p_{\{R|X_l=0\}}(x) \,||\, p_R)$$

$$FLD(X_l : R \,|\, f) = f(LD(X_l : R)) \qquad f : (0, \infty) \rightarrow \Re^+$$

(f is a convex function)

$$FLD(X_l : R \,|\, f) = LD(X_l : R) \qquad f = x$$

$$FLD(X_l : R \,|\, f) = e^{LD(X_l:R)} - 1 \qquad f = e^x - 1$$

Family of latent divergences is convex

Ranking Algorithm

Algorithm 1 Ranking Algorithm

Require: CodeCoverageMatrix: X, ResultVector: R
Ensure: Lines are ranked.

$M \leftarrow \text{GetNumberOfColumns}(X)$
for $l = 1$ to M **do**
  $Y[l] \leftarrow \mathcal{LD}(X_l : R)$
end for
{Normalize the array Y to capture the ranks; the lines having with maximum latent divergence will be ranked 1.}
for $i = 1$ to $|Y|$ **do**
  $Z[i] \leftarrow \frac{Y[i]}{max(Y)}$
end for
for $i = 1$ to $|Z|$ **do**
  $Q[i] \leftarrow \lfloor Z[i] + M \times (1 - Z[i]) \rfloor$
end for

Metric Quality

How good is a metric at localizing faults?

- Rank irrelevant lines of code lower
- Rank relevant lines of code higher

Our measure:

$$\phi = \sum_{i=1}^{M} \left(\frac{1}{Q(i) \times M} \right)$$

Outline

Overview

Example

Framework

Experiments

Conclusion

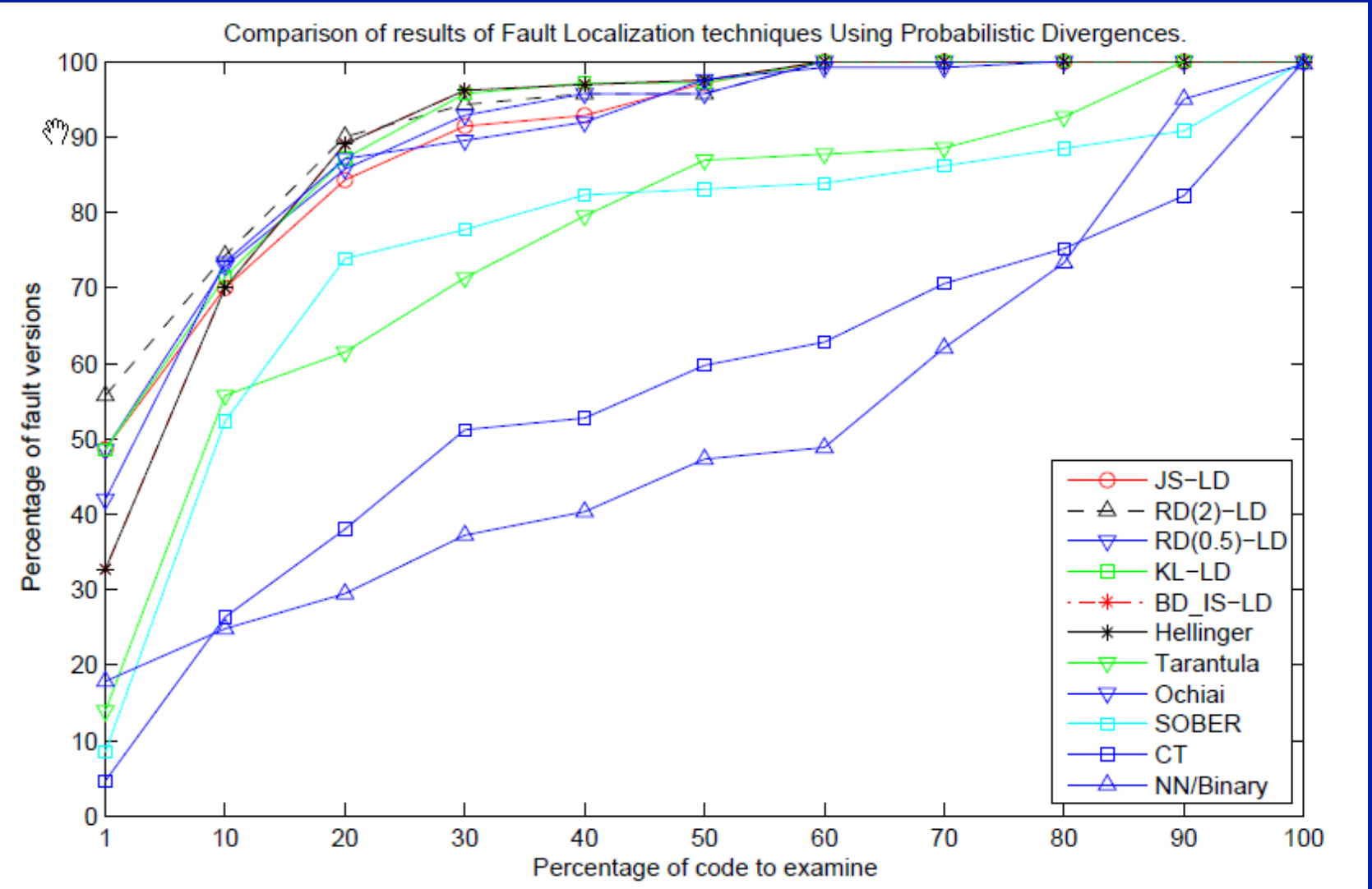
Subject programs

Siemens suite

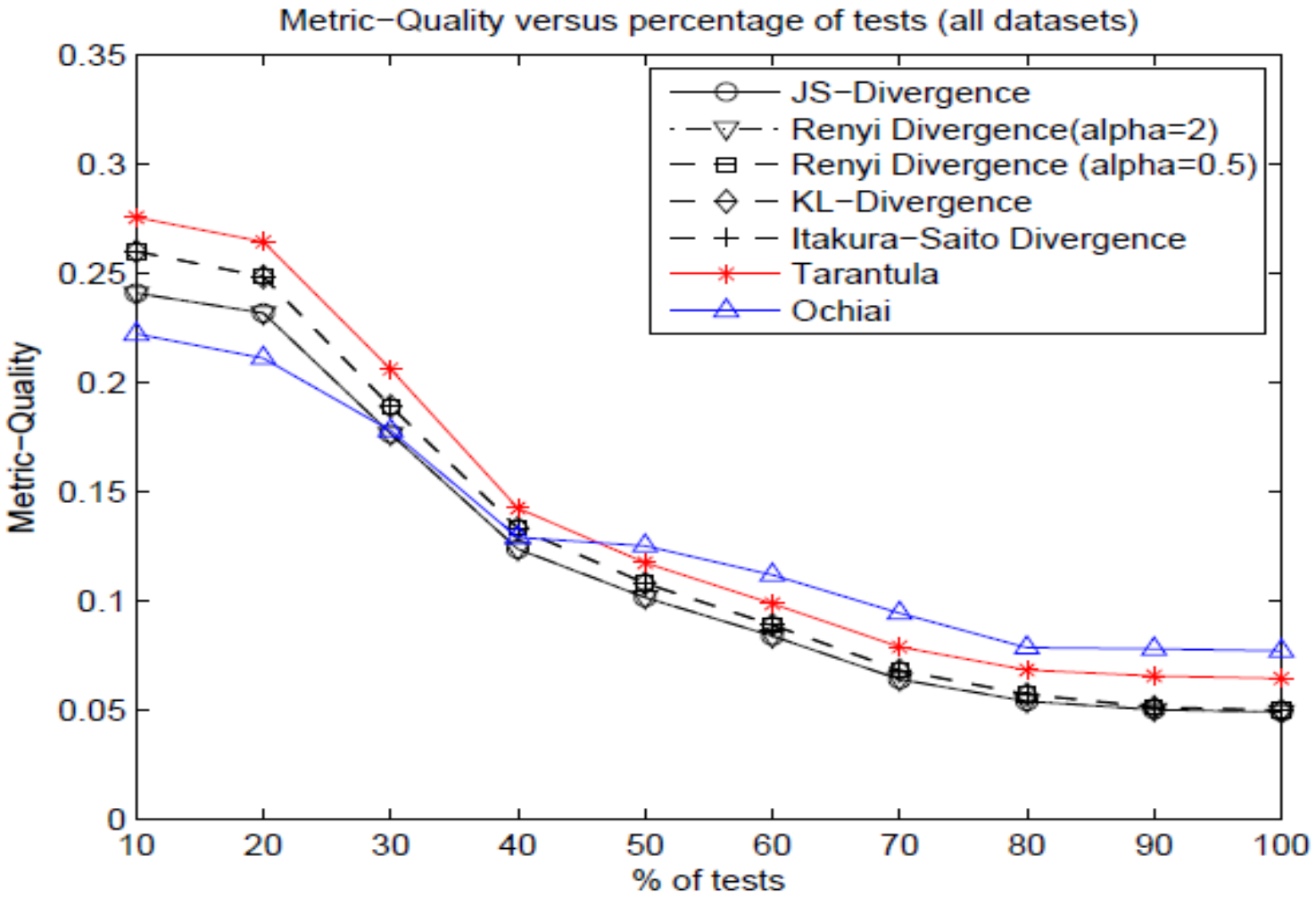
- Widely used in SE research

Program	Description	Versions	LOC	Executable	Testcases
print_tokens	Lexical Analyser	7	565	175	4130
print_tokens2	Lexical Analyser	10	510	178	4115
replace	Pattern replacement	32	563	216	5542
schedule	Priority Scheduler	9	412	121	2650
schedule2	Priority Scheduler	10	307	112	2710
tcas	Altitude Separation	41	173	55	1608
tot-info	Information Measure	23	406	113	1052

Results: Fault Localization Effectiveness



Results: Metric Quality



Outline

Overview

Example

Framework

Experiments

Conclusion

Related Work

Fault localization

- Program-spectra
  - Nearest neighbor [Renieris+]
  - Tarantula [Jones+]
  - Ochiai [Abreu+]
- Memory graph
  - Delta debugging [Zeller]
  - Cause transition [Cleve+]
- Program predicates
  - Statistical debugging [Liblit+]
  - SOBER [Liu+]

Program repair [Weimer+, Malik+, Jeffrey+, Wei+, Gopinath+]

? & //

This paper presents a novel approach to fault localization using latent divergence

- Based on probabilistic divergence

Key insight: fault localization reduces to feature selection

Experimental results show effectiveness of our approach

We believe machine learning techniques have a fundamental role in increasing our ability to deploy reliable code

- Program repair
- Runtime checking
- Failure clustering

sroychow | khurshid@ece.utexas.edu