Evolving Fault Localisation

Shin Yoo, University College London, UK

Human Competitive Award, GECCO 2012, 09 July 2012
Spectra Based Fault Localisation
Spectra Based Fault Localisation

Program

Tests
Spectra Based Fault Localisation
Spectra Based Fault Localisation

Program $\rightarrow$ Spectrum $\rightarrow$ Risk Evaluation Formula

$e_f - \frac{e_p}{e_p + n_p + 1}$
Spectra Based Fault Localisation

Program

\[ e_f - \frac{e_p}{e_p + n_p + 1} \]

Spectrum

Risk Evaluation Formula

Tests

Ranking
Evolving SBFL

Program → Spectrum → $e_f - \frac{e_p}{e_p + n_p + 1}$ → Ranking

Tests
Evolving SBFL

Risk Evaluation Formula

\[ e_f = \frac{e_p}{e_p + n_p + 1} \]

Ranking
Evolving SBFL

Program -> Spectrum -> GP

Ranking
Evolving SBFL
Evolving SBFL
Evolving SBFL

Fitness

\[ e_f^2(2e_p + 2e_f + 3n_p) \]

\[ e_f^2(e_f^2 + \sqrt{n_p}) \]

\[ \ldots \]
State of the Art

Over 30 formulæ in the literature, with various empirical studies with slightly different results
State of the Art

Over 30 formulæ in the literature, with various empirical studies with slightly different results
State of the Art

Optimality Proof (Naish et al. 2011)

\[ Op1 = \begin{cases} 
-1 & \text{if } n_f > 0 \\
np & \text{otherwise}
\end{cases} \quad Op2 = e_f - \frac{e_p}{e_p + np + 1} \]
State of the Art

Optimality Proof (Naish et al. 2011)

\[
\begin{align*}
Op_1 &= \begin{cases} 
-1 & \text{if } n_f > 0 \\
\frac{n_f}{n_p} & \text{otherwise}
\end{cases} \\
Op_2 &= e_f - \frac{e_p}{e_p + n_p + 1}
\end{align*}
\]

But the proof is against a specific model

```c
if (t1())
    s1(); /* S1 */
else
    s2(); /* S2 */
if (t2())
    x = True; /* S3 */
else
    x = t3(); /* S4 - BUG */
```
State of the Art

Optimality Proof (Naish et al. 2011)

\[ Op_1 = \begin{cases} 
-1 & \text{if } n_f > 0 \\
np & \text{otherwise}
\end{cases} \quad Op_2 = e_f - \frac{ep}{ep + np + 1} \]

But the proof is against a specific model

```c
if (t1())
    s1(); /* S1 */
else
    s2(); /* S2 */
if (t2())
    x = True; /* S3 */
else
    x = t3(); /* S4 - BUG */
```

... not to mention hard.
Human Competitiveness

How many of 9 Existing Techniques can 30 GP runs match and/or outperform?

- 6 runs outperform 8 existing techniques and match/outperform one of the state of the art with proof (Op1 and Op2).
- 16 runs outperform all 7 existing techniques without proof.

Four Unix tools with 92 faults: 20 random faults for training, 72 for evaluation.
Human Competitiveness

- Per-fault view shows that evolved techniques can outperform ones with optimality proofs.
Human Competitiveness

- Per-fault view shows that evolved techniques can outperform ones with optimality proofs.
Human Competitiveness

・Per-fault view shows that evolved techniques can outperform ones with optimality proofs.
Future of Search-Based Software Engineering

From Solutions to Generic Problems... To Techniques and Strategies for Your Problems.
The most effective way to do it, is to do it.

- GP provides a structured, automated way of doing iterative design.
- It can cope with a much diverse spectra and other meta-data.
- GP can evolve to suit your project.
The most effective way to do it, is to do it.

- GP provides a structured, automated way of doing iterative design.
- It can cope with a much diverse spectra and other meta-data.
- GP can evolve to suit your project.
The most effective way to do it, is to do it.

- GP provides a structured, automated way of doing iterative design.
- It can cope with a much diverse spectra and other meta-data.
- GP can evolve to suit your project.
The most effective way to do it, is to do it.

- GP provides a structured, automated way of doing iterative design.
- It can cope with a much diverse spectra and other meta-data.
- GP can evolve to suit your project.

Diagram:
- Spectrum
  - Think Hard
    - Write Formula
      - Experiment
        - Human
    - Evaluate
      - Select
        - GP
The most effective way to do it, is to do it.

• GP provides a structured, automated way of doing iterative design.

• It can cope with a much diverse spectra and other meta-data.

• GP can evolve to suit your project.
The most effective way to do it, is to do it.

- GP provides a structured, automated way of doing iterative design.
- It can cope with a much diverse spectra and other meta-data.
- GP can evolve to suit your project.
The most effective way to do it, is to do it.

- GP provides a structured, automated way of doing iterative design.
- It can cope with a much diverse spectra and other meta-data.
- GP can evolve to suit your project.
Evolving SBFL

Risk Evaluation Formula

$$e_f = \frac{e_p}{e_p + n_p + 1}$$

Program → Spectrum

Tests ↓ Ranking
Evolving SBFL

Risk Evaluation Formula

\[ e_f = \frac{e_p}{e_p + n_p + 1} \]

Program → Spectrum → Risk Evaluation Formula → Ranking

Tests

Human Competitiveness

How many of 9 Existing Techniques can 30 GP runs match and/or outperform?

- 6 runs outperform 8 existing techniques and match/outperform one of the state of the art with proof (Op1 and Op2).
- 16 runs outperform all 7 existing techniques without proof.

Four Unix tools with 92 faults: 20 random faults for training, 72 for evaluation.
Evolving SBFL

\[ e_f = \frac{e_p}{e_p + n_p + 1} \]

Risk Evaluation Formula

Program → Spectrum → Ranking

Tests

The most effective way to do it, is to do it.

* GP provides a structured, automated way of doing iterative design.

* It can cope with a much diverse spectra and other meta-data.

* GP can evolve to suit your project.

Human Competitiveness

How many of 9 Existing Techniques can 30 GP runs match and/or outperform?

* 6 runs outperform 8 existing techniques and match/outperform one of the state of the art with proof (Op1 and Op2).

* 16 runs outperform all 7 existing techniques without proof.

Four Unix tools with 92 faults: 20 random faults for training, 72 for evaluation.
**Evolving SBFL**

Program → Spectrum → \( e_f = \frac{e_p}{e_p + np + 1} \) → Risk Evaluation Formula → Ranking → Tests

**Human Competitiveness**

How many of 9 Existing Techniques can 30 GP runs match and/or outperform?

- 6 runs outperform 8 existing techniques and match/outperform one of the state of the art with proof (Op1 and Op2).
- 16 runs outperform all 7 existing techniques without proof.

Four Unix tools with 92 faults: 20 random faults for training, 72 for evaluation.

**The most effective way to do it, is to do it.**

- GP provides a structured, automated way of doing iterative design.
- It can cope with a much diverse spectra and other meta-data.
- GP can evolve to suit your project.

**Detailed Statistics & Spectra Data**

http://www.cs.ucl.ac.uk/staff/s.yoo/evolving-sbfl.html