Search-Based Software Maintenance and Evolution

Annibale Panichella

Advisor:
Prof. Andrea De Lucia
Dott. Rocco Oliveto

4/16/2014
Search-Based Software Engineering

«The application of meta-heuristic search-based optimization techniques to find near-optimal solutions in software engineering problems.»

1. **Problem Reformulation**: reformulating typical SE problems as optimization problems

2. **Fitness Function**: definition of functions to optimize

3. **Optimization Algorithms**: applying search algorithm to solve such functions
   - Genetic Algorithms
   - Hill climbing
   - Simulated Annealing
   - Random Search
   - Tabu Search
   - Particle Swarm Optimization
   - ...
Why SBSE?

Large Search Space

Presence of conflicting goals
Optimization Problem

\[ \min f(x) = \sin((x-1)^3) + 1 \]
Genetic Algorithms (GAs)

\[ \min f(x) = \sin((x - 1)^8) + 1 \]
Genetic Algorithms (GAs)

\[
\min f(x) = \sin((x - 1)^8) + 1
\]
Genetic Algorithms (GAs)

\[
\min f(x) = \sin((x - 1)^8) + 1
\]
Genetic Algorithms (GAs)

\[
\min f(x) = \sin((x - 1)^8) + 1
\]
Genetic Algorithms (GAs)

\[
\min f(x) = \sin((x - 1)^8) + 1
\]
Genetic Algorithms (GAs)

\[
\min f(x) = \sin((x - 1)^8) + 1
\]

![Graph showing the function \( f(x) = \sin((x - 1)^8) + 1 \)]
Genetic Algorithms (GAs)

\[
\min f(x) = \sin((x - 1)^8) + 1
\]
Genetic Algorithms (GAs)

\[
\min f(x) = \sin((x - 1)^8) + 1
\]
Genetic Algorithms (GAs)

$$\min f(x) = \sin((x - 1)^8) + 1$$

Initial Population

Crossover

Mutation

Selection

No

End?

Yes
Genetic Algorithms (GAs)

\[ \min f(x) = \sin((x - 1)^8) + 1 \]
Main Contributions

Search-Based Program Comprehension

Multi-Objectives Defect Prediction

Search-Based Test Data Generation

Multi-Objective Test Suite Optimization
Main Contributions

Search-Based Program Comprehension

Multi-Objectives Defect Prediction

Search-Based Test Data Generation

Multi-Objective Test Suite Optimization
**Program Comprehension**

```java
public class LoadConfiguration extends AbstractHandler {

IWorkbench wb = PlatformUI.getWorkbench();
IWorkbenchWindow window = wb.getActiveWorkbenchWindow();

public LoadConfiguration() {
}

@Override
public Object execute(ExecutionEvent event) throws ExecutionException {
IWorkbench wb = PlatformUI.getWorkbench();
IWorkbenchWindow window = wb.getActiveWorkbenchWindow();

IWorkbenchPage page = window.getActivePage();
IEditorPart editor = page.getActiveEditor();

// reading the instantiation variable in SCM
ResourceSet resourceSet = new ResourceSetImpl();

IFile file1;
try {
    file1 = (IFile) editor.getEditorInput().getAdapter(IFile.class);
} catch (Exception exc) {
    printError("Please select a State Chart Model", window);
    return null;
}

SCMDiagram scd = null;
Resource scdResource = resourceSet.createResource();
try {
    scdResource.load(null);
    scd = (SCMDiagram) scdResource.getContents().get(0);
} catch (IOException e) {
    printError("Corrupted State Chart Model file", window);
    scdResource = null;
    return null;
}
```

"Software that is not comprehended cannot be changed" - Rajlich and Wilde - ICPC 2002

40% to 60% of the maintenance effort is devoted to understanding the software to be modified - Dorfman and Thayer – IEEE Software Engineering 1996
Information Retrieval

Artefacts Indexing

Software Artefacts → Term Extraction → Stop Word Removal → Morphological Analysis → Term Weighting

Similarity Computation

Distance Function → IR Model

Software Maintenance task:
- Traceability Recovery
- Source code labeling
- Bug duplication
- Feature Location
- ...

Search-Based Software Maintenance and Testing
Information Retrieval

Software Artefacts

Term Extraction
- Special Chars.
- Digits
- White spaces ...

Stop Word Removal
- Stop-word function
- Java stop-word list
- English stop-word list
- Italian stop-word list ...

Morphological Analysis
- No Stemmer
- Porter Stemmer
- English Snowball Stemmer
- Italian Snowball Stemmer ...

Term Weighting
- Boolean tf
- tf-idf
- Log(tf+1)
- Entropy ...

Distance Function
- Cosine Similarity
- Jaccard Distance
- Euclidean Distance
- Jensen-Shannon Divergence ...

IR Model
- Vector Space Model
- Latent Semantic Indexing
- Latent Dirichlet Allocation
- Jensen-Shannon Divergence ...

Software Maintenance task:
- Traceability Recovery
- Source code labeling
- Bug duplication
- Feature Location
- ...

Search-Based Software Maintenance and Testing
What is the right IR process?

It is not possible to build a set of guidelines for assembling IR-based solutions for a given data set.

Different dataset require different IR parameters.

# Predicting the performances?

| Term Extraction | Special Chars.  
|                 | Digits  
|                 | White space  
| Stop Word Removal | Stop-word function  
|                 | Java stop-word list  
|                 | English stop-word list  
|                 | Italian stop-word list  
| Morphological Analysis | No Stemmer  
|                 | Porter Stemmer  
|                 | English Snowball Stemmer  
|                 | Italian Snowball Stemmer  
| Term Weighting | Boolean  
|                 | tf  
|                 | tf-idf  
|                 | Log(tf+1)  
|                 | Entropy  
| IR Model | LSI (k)  
|                 | LDA (alpha, beta, n, k)  
| Distance Function | Cosine Similarity  
|                 | Hellinger Distance  

4/16/2014

Search-Based Software Maintenance and Testing
Predicting the performances?

- **Term Extraction**
  - Special Chars.
  - Digits
  - White space

- **Stop Word Removal**
  - Stop-word function
  - Java stop-word list
  - English stop-word list
  - Italian stop-word list

- **Morphological Analysis**
  - No Stemmer
    - Porter Stemmer
    - English Snowball Stemmer
    - Italian Snowball Stemmer

- **Term Weighting**
  - Boolean
  - tf
  - tf-idf
  - Log(tf+1)
  - Entropy

- **IR Model**
  - LSI (k=3)
  - LDA (alpha, beta, n, k)

- **Distance Function**
  - Cosine Similarity
  - Hellinger Distance
Predicting the performances?

- **Term Extraction**
  - Special Chars.
  - Digits
  - White space

- **Stop Word Removal**
  - Stop-word function
  - Java stop-word list
  - English stop-word list
  - **Italian stop-word list**

- **Morphological Analysis**
  - No Stemmer
  - Porter Stemmer
  - **English Snowball Stemmer**
  - Italian Snowball Stemmer

- **Term Weighting**
  - Boolean
  - tf
  - tf-idf
  - Log(tf+1)
  - Entropy

- **IR Model**
  - LSI (k=4)
  - LDA (alpha, beta, n, k)

- **Distance Function**
  - Cosine Similarity
  - Hellinger Distance

4/16/2014
Search-Based Software Maintenance and Testing
### Term Extraction
- **Special Chars.**
  - Digits
  - White space

### Stop Word Removal
- Stop-word function
  - Java stop-word list
  - English stop-word list
  - Italian stop-word list

### Morphological Analysis
- No Stemmer
- Porter Stemmer
- English Snowball Stemmer
- **Italian Snowball Stemmer**

### Term Weighting
- Boolean
- **tf**
- **tf-idf**
- Log(tf+1)
- Entropy

### IR Model
- LSI (k=4)
- LDA (alpha, beta, n, k)

### Distance Function
- **Cosine Similarity**
- Hellinger Distance

---

**Predicting the performances?**

![Diagram showing term 1 and term 2 with documents Doc 1 and Doc 2]
Conjecture: there is a *relationship* between *quality of clusters* and IR process *performances*
Search-Based Solution (LSI-GA)

1) **Problem Reformulation**: Finding the IR process which maximize the quality of clusters

2) **Solution Encoding**

   \[
   X = \text{Char. Pruning, Identifier Splitting, Stop word removal, Morph. Analysis, Term Weighting, IR Technique Settings}
   \]

3) **Fitness Function**: Silhouette Coefficient

   \[
   F(X) = \text{Silhouette Coefficient} (X) = \frac{1}{n} \sum_{i=1}^{n} \frac{\text{separation}(d_i) - \text{cohesion}(d_i)}{\max\{\text{separation}(d_i), \text{cohesion}(d_i)\}}
   \]

4) **Solver**: Genetic Algorithms
Empirical Evaluation

1) Traceability Recovery

<table>
<thead>
<tr>
<th>System</th>
<th>Artifacts</th>
<th>Type</th>
<th>Number</th>
<th>Total</th>
<th>N. Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>EasyClinic</td>
<td>Use Case</td>
<td>30</td>
<td></td>
<td>77</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Code Classes</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eTour</td>
<td>Use Case</td>
<td>58</td>
<td></td>
<td>174</td>
<td>246</td>
</tr>
<tr>
<td></td>
<td>Code Classes</td>
<td>116</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iTrust</td>
<td>Code Classes</td>
<td>33</td>
<td></td>
<td>149</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>JSP</td>
<td>116</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2) Feature Location

<table>
<thead>
<tr>
<th>System</th>
<th>KLOC</th>
<th>Files</th>
<th>Methods</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>jEdit</td>
<td>104</td>
<td>503</td>
<td>6,413</td>
<td>159</td>
</tr>
<tr>
<td>JabReg</td>
<td>74</td>
<td>579</td>
<td>4,607</td>
<td>39</td>
</tr>
</tbody>
</table>

3) Bug Report Duplication

<table>
<thead>
<tr>
<th>System</th>
<th>N. Bug Rep.</th>
<th>N. Duplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse</td>
<td>225</td>
<td>44</td>
</tr>
</tbody>
</table>
# Empirical Evaluation

1) **Traceability Recovery**

<table>
<thead>
<tr>
<th>System</th>
<th>Artifacts</th>
<th>Type</th>
<th>Number</th>
<th>Total</th>
<th>N. Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>EasyClinic</td>
<td>Use Case</td>
<td>30</td>
<td></td>
<td>77</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Code Classes</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eTour</td>
<td>Use Case</td>
<td>58</td>
<td></td>
<td>174</td>
<td>246</td>
</tr>
<tr>
<td></td>
<td>Code Classes</td>
<td>116</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iTrust</td>
<td>Code Classes</td>
<td>33</td>
<td></td>
<td>149</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>JSP</td>
<td>116</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1) **Traceability Recovery**

<table>
<thead>
<tr>
<th>System</th>
<th>Artifacts</th>
<th>Type</th>
<th>Number</th>
<th>Total</th>
<th>N. Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>EasyClinic</td>
<td>Use Case</td>
<td>30</td>
<td>77</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Code Classes</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eTour</td>
<td>Use Case</td>
<td>58</td>
<td>174</td>
<td>246</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Code Classes</td>
<td>116</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iTrust</td>
<td>Code Classes</td>
<td>33</td>
<td>149</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JSP</td>
<td>116</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Experimented techniques:**

1. LSI-GA
2. Previously published IR process
3. Ideal IR process

**Performance metrics:**

*Average precision*
Results

LSI-GA outperforms baseline (p-value < 0.05)
Ideal is statistically better than LSI-GA
Configuring LDA using GAs


How to Effectively Use Topic Models for Software Engineering Tasks? An Approach Based on Genetic Algorithms

A significant amount of research on applying Information Retrieval (IR) methods for analyzing technical information in software artifacts [2] has been conducted in the SE community in recent years. Among the popular and promising IR techniques, we emphasize Latent Semantic Indexing (LSI) [2] and Latent Dirichlet Allocation (LDA) [3]. The latter is a probabilistic statistical model that estimates distributions of latent topics from textual documents. It assumes that these documents have been generated using the probability distribution of these topics, and that the words in the documents were generated probabilistically in a similar manner.

A number of approaches using LSI and LDA have been proposed to support software engineering tasks such as feature location [4], change impact analysis [5], bug localization [6], code detection [7], maintainability link recovery [8], error developer recommendation [9], code structure recovery [11], [12], dependency analysis [13], and others [14], [15], [16]. In all these approaches, LDA and LSI have been used on textual information extracted from natural language documents (i.e., the same settings, parameters, and parameters for the underlying assumption was that source code for other software artifacts and natural language documents exhibit similar properties. Most specifically, applying LDA requires setting the number of topics and other parameters specific to the particular LDA implementation. For example, the fast collapsed Gibbs sampling generative model for LDA requires setting the number of iterations and the Dirichlet distribution parameters α and β [17]. Even though LDA was successfully used in the IR and natural language analysis community, applying it on software data, using the same parameter values used for natural language text, did not always produce the expected results [18]. As is the case of machine learning and optimization techniques, a poor parameter calibration or setting assumptions about the nature of the data could lead to poor results [19].

Recent research has challenged this assumption and showed that text extracted from source code is much more representative of the source code itself than human natural language text. According to recent empirical findings, "corpus-based statistical language models capture [20] a much higher level of regularity in software, even more so than in English" [20]. This fundamental new research finding implies that only those highly sophisticated IR methods showed rather low performance when applied to software data using parameters and configurations that were generally applicable for and tested on natural language corpora.

This paper builds on the finding that text in software artifacts has different properties, as compared to natural language text, thus, we need new solutions for characterizing and configuring LDA and LSI to achieve better (acceptable) performance on software engineering tasks. This paper introduces LDA-GA,
Other works on P.C.


Other works on P.C.


Main Contributions

- Search-Based Program Comprehension
- Search-Based Test Data Generation
- Multi-Objectives Defect Prediction
- Multi-Objective Test Suite Optimization
Bugs are everywhere...
Practical Constraints

- Software Quality
- Money Time
Defect Prediction

Spent more resources on components most likely to fail
Defect Prediction Methodology

Past Projects → Predictors → Past Defects → Predicting Model → Defect Prone

New Project → Predictors

Class1: YES
Class2: YES
Class3: NO
...: YES
ClassN: ...

Past Defects

Search-Based Software Maintenance and Testing

4/16/2014
Defect Prediction Methodology

All the existing predicting models work on precision and not on cost

We need COST-oriented models
Multi-objective Defect Prediction
Multi-objective Reformulation

1) **Problem Reformulation**: Finding the logistic regression coefficients \((a, b, c, \ldots)\) that optimize cost and effectiveness

\[
\text{Logit} = \frac{e^{a + b m_{i1} + c m_{i2} + \ldots}}{1 + e^{a + b m_{i1} + c m_{i2} + \ldots}}
\]

2) **Objectives Function**:

\[
\begin{align*}
\min \quad & \text{Cost} = \sum_i \text{Pred}_i \cdot \text{LOC}_i \\
\max \quad & \text{Recall} = \sum_i \text{Pred}_i \cdot \text{Bug}_i
\end{align*}
\]

3) **Solver**: Multi-objective Genetic Algorithms (NSGA-II)
Multi-objective Genetic Algorithm

Pareto Optimality: all solutions that are not dominated by any other solutions form the Pareto optimal set

Multiple objectives are optimized using Pareto efficient approaches

Multiple optimal solutions (models) can be found

Recall/Effectiveness

Cost
Empirical Evaluation

Context:

<table>
<thead>
<tr>
<th>Name</th>
<th># Classes</th>
<th># Defect-Prone Classes</th>
<th>% Defect-Prone Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant</td>
<td>745</td>
<td>166</td>
<td>22%</td>
</tr>
<tr>
<td>Camel</td>
<td>965</td>
<td>188</td>
<td>19%</td>
</tr>
<tr>
<td>Ivy</td>
<td>352</td>
<td>40</td>
<td>11%</td>
</tr>
<tr>
<td>jEdit</td>
<td>306</td>
<td>75</td>
<td>25%</td>
</tr>
<tr>
<td>Log4j</td>
<td>205</td>
<td>189</td>
<td>92%</td>
</tr>
<tr>
<td>Lucene</td>
<td>340</td>
<td>203</td>
<td>60%</td>
</tr>
<tr>
<td>Poi</td>
<td>442</td>
<td>281</td>
<td>64%</td>
</tr>
<tr>
<td>Prop</td>
<td>661</td>
<td>66</td>
<td>10%</td>
</tr>
<tr>
<td>Tomcat</td>
<td>858</td>
<td>77</td>
<td>9%</td>
</tr>
<tr>
<td>Xalan</td>
<td>910</td>
<td>898</td>
<td>99%</td>
</tr>
</tbody>
</table>

Experimented Algorithms:

1. Multi-objective cross-project Logistic Regression
2. Traditional cross-project Logistic Regression
3. Traditional within-project Logistic Regression
4. Clustering (local) cross-project defect prediction

Performance metrics:
Cost = # LOC to analyze
Effectiveness/Recall = % defect-prone classes identified
Results

jEdit

Log4j

Multi-Objective Logistic  Single-Objective Logistic  Clustering Logistic  Within Project Logistic
Multi-Objective Defect Prediction

G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella

Multi-Objective Cross-Project Defect Prediction. ICST 21013

Defect Prediction as a Multi-Objective Optimization Problem

G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella

Defect Prediction as Multi-Objective Optimization Problem.
Submitted as Special Issue on Journal STVR
Main Contributions

Search-Based Program Comprehension

Multi-Objectives Defect Prediction

Search-Based Test Data Generation

Multi-Objective Test Suite Optimization
Diversity is essential to the genetic algorithm because it enables the algorithm to search a larger region of the space.
GAs in Software Testing

Diversity is essential to the genetic algorithm because it enables the algorithm to search a larger region of the space.
GAs in Software Testing

Diversity is essential to the genetic algorithm because it enables the algorithm to search a larger region of the space.

Population drift
public class Triangle {

    public String check (double a, double b, double c) {

        if (a == b) {
            if (a == c)
                return 'equilater';
            else
                return 'isoscele';
        } else {
            if (a == c || b == c)
                return 'isoscele';
            else
                return 'scalene';
        }
    }
}
public class Triangle {

    public String check (double a, double b, double c){

        1. if(a == b){
        2.     if(a == c)
            3.         return 'equilater';
        4.     else
            5.         return 'isoscele';
        6.     } else{
            7.         if(a == b || a == c || b == c)
                8.             return 'isoscele';
            9.         else
                10.             return 'scalene';

        }

    }

}
Search-based approach

public class Triangle {

    public String check (double a, double b, double c) {
        if (a == b) {
            if (a == c) {
                return 'equilateral';
            } else {
                return 'isoscele';
            }
        } else {
            if (a == b || a == c || b == c) {
                return 'isoscele';
            } else {
                return 'scalene';
            }
        }
    }
}

Branch distance

\[
\min f(a,b,c) = 2 \times \text{abs}(a - b) + \text{abs}(a - c) \]

Search-based approach

Search-Based Software Maintenance and Testing
public class Triangle {

    public String check (double a, double b, double c){

        if(a == b) {
            if(a == c) return 'equilater';
            else return 'isoscele';
        }
        else {
            if(a == b || a == c || b == c)
                return 'isoscele';
            else return 'scalene';
        }
    }
}

Branch distance

(a == b) -> abs(a - b)
(a == c) -> abs(a - c)

min f(a,b,c) = 2 * abs(a - b) + + abs(a - c)

Test Case 4
Triangle t= new Triangle();
String s=t.check(2,2,2)
Triangle Program

c=2 \quad a, b \in [-1;4]

1) Flat search space
2) Several Local optimal
3) Only one global optimum
GAs Simulation

Branch Distance

a, b ∈ [-30;30]
c=2

Mutation Rate = 0.10
Population = 50
Crossover = single-point

Premature convergence (genetic drift)
Injecting Diversity during the Evolution
What is the evolution direction?

\[ P(t) = \text{Population at generation } t \]
What is the evolution direction?

\[ P(t) = \text{Population at generation } t \]

\[ P(t+k) = \text{Population after } k \text{ generations} \]
What is the evolution direction?

$P(t) =$ Population at generation $t$

$P(t+k) =$ Population after $k$ generations

Evolution Directions
Why?

\[ P(t) = \text{Population at generation } t \]

\[ P(t+k) = \text{Population after } k \text{ generations} \]

Evolution Directions

Orthogonal Individuals
How? Singular Value Decomposition

Population at generation $t$

$$P_t = U_t \cdot \Sigma_t \cdot V_t$$

Population at generation $t + k$

$$P_{t+k} = U_{t+k} \cdot \Sigma_{t+k} \cdot V_{t+k}$$

The current evolution direction is proportional to

$$\overline{V} = V_{t+k} - V_t$$

$$\overline{\Sigma} = \Sigma_{t+k} - \Sigma_t$$
Using SVD for Evolution Direction

\[ U_t + \Sigma_t \cdot V_t + \Sigma_t \cdot V^T \cdot P_t \]

Evolution Direction

\[ \Sigma \cdot \tilde{V} \]

\( P_t \)

\( P_{t+k} \)
Using SVD for Evolution Direction

$$U_{t+k} \cdot (\Sigma_{t+k} + \bar{\Sigma}) \cdot (V_{t+k} + \bar{V})^T$$
Then, we construct a new orthogonal population as follows:

\[ U_{t+k} \cdot (\Sigma_{t+k} + \bar{\Sigma}) \cdot (V_{t+k} + \bar{V})^T \]
Integration SVD with Standard GA

1. Initialize population
2. Selection
3. Crossover
4. Mutation

- Rank Scaling Selection
- Single-point crossover
- Uniform mutation

Terminate?
SVD + Standard GAs

Initialize population

Selection

Crossover

Mutation

Select best 50% of individuals

Generate an orthogonal sub-population

Replace the worst 50% of individuals with new sub-populations

Terminate?

No

Yes
Simulation on Triangle Program

Standard GA

SVD-GA
# Empirical study

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Coverage Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>ArithmeticUtils</td>
<td>99</td>
</tr>
<tr>
<td>P2</td>
<td>Arrays</td>
<td>75</td>
</tr>
<tr>
<td>P3</td>
<td>Beta</td>
<td>90</td>
</tr>
<tr>
<td>P4</td>
<td>CreditCardValidator</td>
<td>32</td>
</tr>
<tr>
<td>P5</td>
<td>Complex</td>
<td>126</td>
</tr>
<tr>
<td>P6</td>
<td>FastMath</td>
<td>60</td>
</tr>
<tr>
<td>P7</td>
<td>Fraction</td>
<td>108</td>
</tr>
<tr>
<td>P8</td>
<td>IPAddressValidator</td>
<td>243</td>
</tr>
<tr>
<td>P9</td>
<td>LUDecomposition</td>
<td>76</td>
</tr>
<tr>
<td>P10</td>
<td>KolmogorovDistribution</td>
<td>50</td>
</tr>
<tr>
<td>P11</td>
<td>QRDecomposition</td>
<td>72</td>
</tr>
<tr>
<td>P12</td>
<td>Quadratic</td>
<td>7</td>
</tr>
<tr>
<td>P13</td>
<td>RootsOfUnity</td>
<td>27</td>
</tr>
<tr>
<td>P14</td>
<td>SaddlePointExansion</td>
<td>16</td>
</tr>
<tr>
<td>P15</td>
<td>Sort</td>
<td>70</td>
</tr>
<tr>
<td>P16</td>
<td>Tomorrow</td>
<td>107</td>
</tr>
<tr>
<td>P17</td>
<td>TriangularDistribution</td>
<td>50</td>
</tr>
</tbody>
</table>

**Experimented Algorithms:**
1. SVD-GA
2. R-GA
3. R-SVD-GA
4. Standard GA

**Performance metrics:**
- Effectiveness = % covered branches
- Efficiency/cost = # executed statements
**RQ1:** Does orthogonal exploration improve the effectiveness of evolutionary test case generation?
RQ2: Does orthogonal exploration improve the efficiency of evolutionary test case generation?
Orthogonal exploration

GECCO 2012

Orthogonal Exploration of the Search Space in Evolutionary Test Case Generation F. M. Kifetew, A. Panichella, A. De Lucia, R. Oliveto, P. Tonella
ISSTA 2013
Main Contributions

Search-Based Program Comprehension

Multi-Objectives Defect Prediction

Search-Based Test Data Generation

Multi-Objective Test Suite Optimization
Software Evolution

Software continuously changes (evolves):
• Add new functionalities
• Removing old functionalities
• Bug fixing activities
• ...

Time
Regression Testing

Software before changes

- Test Case 1
- Test Case 2
- Test Case 3
- ...
- Test Case n

Software after changes

- Test Case 1
- Test Case 2
- Test Case 3
- ...
- Test Case n
Regression Testing is time consuming

1000 machine-hours to execute 30,000 functional test cases for a software product...

Mirarab, et al. The effects of time constraints on test case prioritization: A series of controlled experiments. TSE 2010
Test Suite Optimization

- Test Case 4: Ignore
- Test Case 3: Run
- Test Case 2: Run
- Test Case 1: Ignore

Graph:
- Code Coverage: maximize
- Execution Cost: minimize
Multi-Criteria Regression Testing

Multiple objectives are optimized using Pareto efficient approaches

Multiple optimal solutions can be found

Pareto Optimality: all solutions that are not dominated by any other solutions form the Pareto optimal set.
There is no clear winner
Multi-Criteria Regression Testing

There is no clear winner

Population Drift
Multi-Criteria Regression Testing

There is no clear winner.

Can we do better?
Diversity Injection in NSGA-II

- Non Dominated Sorting Algorithm
- Crowding Distance
- Tournament Selection

- Multi-points crossover
- Bit-flip mutation

Flowchart:
- Initialize population
- Selection
- Crossover
- Mutation
- Terminate?
  - Yes
  - No

4/16/2014
Search-Based Software Maintenance and Testing
• Use **orthogonal design** methodology to generate well diversified initial population
SVD + NSGA-II

- Use **orthogonal design** methodology to generate well diversified initial population

- **Generate an orthogonal initial population**
  - **Selection**
  - **Crossover**
  - **Mutation**

- **Select best 50% of individuals**
  - **Generate an orthogonal sub-population**
  - **Replace the worst 50% of individuals with new sub-populations**

- **Terminate?**

  - **Yes**
  - **No**
Empirical Evaluation

Software systems:

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>LOC</th>
<th>Test Suite Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bash</td>
<td>59,846</td>
<td>1,200</td>
</tr>
<tr>
<td>2</td>
<td>flex</td>
<td>10,459</td>
<td>567</td>
</tr>
<tr>
<td>3</td>
<td>grep</td>
<td>10,068</td>
<td>808</td>
</tr>
<tr>
<td>4</td>
<td>gzip</td>
<td>5,680</td>
<td>215</td>
</tr>
<tr>
<td>5</td>
<td>printtokens</td>
<td>726</td>
<td>4,130</td>
</tr>
<tr>
<td>6</td>
<td>printtokens2</td>
<td>520</td>
<td>4,115</td>
</tr>
<tr>
<td>7</td>
<td>schedule</td>
<td>412</td>
<td>2,650</td>
</tr>
<tr>
<td>8</td>
<td>sechedule2</td>
<td>374</td>
<td>2,710</td>
</tr>
<tr>
<td>9</td>
<td>sed</td>
<td>14,427</td>
<td>360</td>
</tr>
<tr>
<td>10</td>
<td>space</td>
<td>6,199</td>
<td>13,583</td>
</tr>
<tr>
<td>11</td>
<td>vim</td>
<td>122,169</td>
<td>975</td>
</tr>
</tbody>
</table>

Experimented Algorithms:

1. SVD-NSGA-II + Init. Pop
2. NSGA-II
3. Additional Greedy Algorithm

Problems:

1. 2-objectives
   - Execution Cost
   - Code Coverage
2. 3-objectives
   - 2-objectives + Past Faults Coverage

Performance metrics:

# Pareto optimal solutions
% hypervolume = % detected faults per unit time
**Results**

**RQ1:** To what extent does SVD-NSGA-II produce near optimal solutions, compared to alternative techniques?

![Graph showing coverage vs. cost for 'flex' and 'printtokens' datasets]
Results

**RQ1**: To what extent does SVD-NSGA-II produce near optimal solutions, compared to alternative techniques?
RQ2: What is the cost-effectiveness of SVD-NSGA-II compared to the alternative techniques?
Diversity in T.S. Optimization


Summary

Search-Based Program Comprehension

Multi-Objectives Defect Prediction

Search-Based Test Data Generation

Multi-Objective Test Suite Optimization
Thanks!

Question?