Automating Performance Bottleneck Detection using Search-Based Application Profiling

Du Shen, Qi Luo, Denys Poshvyvanyk, Mark Grechanik*

College of William and Mary
*University of Illinois at Chicago

ISSTA 2015
Baltimore, MD, U.S.
1. Introduction

Performance profiling plays a crucial role in software development, allowing programmers to test the efficiency of an application and discover possible performance bottlenecks. Traditional profilers associate performance metrics to nodes or paths of the control flow or call graph by collecting runtime information on specific workloads [2, 19, 27, 39]. These approaches provide valuable information for studying the dynamic behavior of a program and guiding optimizations to portions of the code that take most resources on
Standard application profiling
Standard application profiling

Inputs → Application

- JProfiler
- ADB
- JProbe
- TPTP
- Probekit
- Eclipse
- Netbeans

<table>
<thead>
<tr>
<th>Package</th>
<th>&lt;Base Time (seconds)</th>
<th>Average Base Time (seconds)</th>
<th>Cumulative Time (seconds)</th>
<th>Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.team.collaboration.internal.core.service</td>
<td>249.406940</td>
<td>0.313325</td>
<td>253.997890</td>
<td>796</td>
</tr>
<tr>
<td>com.ibm.team.collaboration.core.service.CollaborationServiceJobQueue</td>
<td>244.485136</td>
<td>12.224257</td>
<td>244.485136</td>
<td>20</td>
</tr>
<tr>
<td>com.ibm.team.collaboration.core.service.CollaborationServiceJob</td>
<td>244.48838</td>
<td>40.747473</td>
<td>244.48838</td>
<td>6</td>
</tr>
<tr>
<td>com.ibm.team.collaboration.core.service.CollaborationServiceJob</td>
<td>0.000133</td>
<td>0.000027</td>
<td>0.0000133</td>
<td>5</td>
</tr>
<tr>
<td>com.ibm.team.collaboration.core.service.CollaborationServiceJob</td>
<td>0.000118</td>
<td>0.000017</td>
<td>0.0000118</td>
<td>7</td>
</tr>
<tr>
<td>com.ibm.team.collaboration.core.service.CollaborationServiceJob</td>
<td>0.000036</td>
<td>0.000036</td>
<td>0.000036</td>
<td>1</td>
</tr>
<tr>
<td>com.ibm.team.collaboration.core.service.CollaborationServiceJob</td>
<td>0.000011</td>
<td>0.000011</td>
<td>0.000011</td>
<td>1</td>
</tr>
</tbody>
</table>
Standard application profiling

1. Agilefant.model.WidgetCollection.getName()  273.2s
2. Agilefant.db.hibernate.UserTypeFilter.deepCopy()  213.5s
3. Agilefant.model.Team.setId()  192.3s
4. Agilefant.model.Backlog.setChildren()  123.9s
5. ......
Standard application profiling

Its success depends on the chosen set of input values

1. Agilefant.model.WidgetCollection.getName() 273.2s
2. Agilefant.db.hibernate.UserTypeFilter.deepCopy() 213.5s
3. Agilefant.model.Team.setId() 192.3s
4. Agilefant.model.Backlog.setChildren() 123.9s
5. ......
find(int list[], int n, int key)
  int lo = 0;
  int hi = n - 1;
  int result = -1;

hi >= lo
return result;

result == -1
hi = mid - 1;
lo = mid + 1;
Result = mid;

final int mid = (lo+hi) / 2;
list[mid] == key
list[mid] > key
lo = mid + 1;
Standard application profiling

```c
int find(int list[], int n, int key)
{
    int lo = 0;
    int hi = n - 1;
    int result = -1;

    while (hi >= lo)
    {
        int mid = (lo + hi) / 2;

        if (list[mid] == key)
        {
            result = mid;
            break;
        }
        else if (list[mid] > key)
        {
            hi = mid - 1;
        }
        else
        {
            lo = mid + 1;
        }
    }

    return result;
}
```
Standard application profiling

It is difficult to choose specific inputs to profile the app and obtain the bottlenecks
Standard application profiling

It is difficult to choose specific inputs to profile the app and obtain the bottlenecks.

It is infeasible to profile the app with all possible combinations of inputs.
Algorithmic Profiling

Dmitrijs Zaparanuks
University of Lugano
Dmitrijs.Zaparanuks@usi.ch

Matthias Hauswirth
University of Lugano
Matthias.Hauswirth@usi.ch

Abstract
Traditional profilers identify where a program spends most of its resources. They do not provide information about why the program spends its resources, where most of the program’s time is spent, or how the program performs on different inputs. In this case, we provide a cost function that relates program input to algorithmic steps. Algorithms researchers and advanced practitioners use cost functions when analyzing the complexity of their algorithms. They usually perform asymptotic analysis to bound cost. In contrast, our algorithmic profiling approach automatically determines approximate values of the considered inputs. However, they may fail to characterize how the performance of a program scales as a function of the input size, which is crucial for the efficiency and reliability of software. Seemingly benign fragments of code may be fast on some testing workloads, passing unnoticed in traditional profilers, while all of these fragments may have a severe impact on the behavior of the program as a whole.

Input-Sensitive Profiling

Emilio Coppa
Dept. of Computer and System Sciences
Sapienza University of Rome
ercoppa@gmail.com

Camil Demetrescu
Dept. of Computer and System Sciences
Sapienza University of Rome
demetres@dis.uniroma1.it

Irene Finocchi
Dept. of Computer Science
Sapienza University of Rome
finocchi@di.uniroma1.it

Abstract
In this paper we present a profiling methodology and toolkit for helping developers discover hidden asymptotic inefficiencies in the code. From one or more runs of a program, our profiler automatically measures how the performance of individual routines scales as a function of the input size.
Input-sensitive profiling

1. Agilefant.model.WidgetCollection.getName() 273.2s
2. Agilefant.db.hibernate.UserTypeFilter.deepCopy() 213.5s
3. Agilefant.model.Team.setId() 192.3s
4. Agilefant.model.Backlog.setChildren() 123.9s
5. ......
Input-sensitive profiling

Inputs

Intellectually intensive and time-consuming

1. Agilefant.model.WidgetCollection.getName() 273.2s
2. Agilefant.db.hibernate.UserTypeFilter.deepCopy() 213.5s
3. Agilefant.model.Team.setId() 192.3s
4. Agilefant.model.Backlog.setChildren() 123.9s
5. ......
Input-sensitive profiling

1 Input x, y, z, u
2 v = A.m(x, y)
3 if (v > z) {
4   C.h(B.m(v))
5 } else {
6   D.h(B.m(v))
7 }
Input-sensitive profiling

1    Input  x, y, z, u
2    v = A.m (x, y)
3    if (v > z) {
4        C.h (B.m (v))
5    }
6    else {
7        D.h (B.m (v))
8    }
Input-sensitive profiling

1    Input x, y, z, u
2    v = A.m(x, y)
3    if (v > z) {
4        C.h(B.m(v))
5    }
6    else {
7        D.h(B.m(v))
8    }

Construct combinations of input values
Input-sensitive profiling

1. Input x, y, z, u
2. v = A.m(x, y)
3. if (v > z) {
4.   C.h(B.m(v))
5. }
6. else {
7.   D.h(B.m(v))
8. }

General-purpose methods
Input-sensitive profiling

1. Input $x, y, z, u$

2. $v = A.m(x, y)$

3. if ($v > z$) {
   
4.     $C.h(B.m(v))$

5. }

6. else {

7.     $D.h(B.m(v))$

8. }

Identify the input-sensitive bottlenecks
Genetic Algorithm-driven Profiler (GA-Prof)

• Automate input-sensitive profiling
• Explore input parameter space
• Detect performance bottlenecks
Genetic Algorithm-driven Profiler (GA-Prof)
Genetic Algorithm-driven Profiler (GA-Prof)

Ranked list:
1. Agilefant.db.hibernate.UserTypeFilter.deepCopy()
2. Agilefant.model.WidgetCollection.getName()
4. Agilefant.model.Team.setId()
5. ..........
Genetic Algorithms (GAs)

• Simulate the natural selection process
• Generate solutions to optimization problems
Genetic Algorithms (GAs)

- Simulate the natural selection process
- Generate solutions to optimization problems
Why do we use GAs in GA-Prof

• Large input space

• Can be formulated as a search and optimization problem
Why do we use GAs in GA-Prof

• Large input space

• Can be formulated as a search and optimization problem

• Performs better than an alternative solution
Why do we use GAs in GA-Prof

- Large input space
- Can be formulated as a search and optimization problem
- Perform better than an alternative solution
Why do we use GAs in GA-Prof

- Large input space
- Can be formulated as a search and optimization problem
- Perform better than an alternative solution

Automatically Finding Performance Problems with Feedback-Directed Learning
Software Testing

GA-Prof vs. FOREPOST
GA Component Definitions

Genes:

Input 1: http://localhost:8080/Agilefant/editUser.action
Input 3: http://localhost:8080/Agilefant/editProduct.action?productId=8
......

A chromosome/individual

Individual 1: 2, 18, 36, 27, 11, 13, 6, 43, 64, 12, 85, 49, 12, 53, 44, 78, 31, 47, 6
Using GAs in GA-Prof

Individual 1
Individual 2
Individual 3
......
Using GAs in GA-Prof

Fitness function:
Individual \leftrightarrow \text{elapsed execution time}
Using GAs in GA-Prof

Fitness function:

Individual 1
Individual 2
Individual 3

......

1. Individual 25  289.5s
2. Individual 17  256.7s
3. Individual 91  197.2s
......

Individual  ❯❯❯ elapsed execution time
Using GAs in GA-Prof

Parent 1: 2, 18, 36, 27, 11, 13, 6, 43, 64, 12, 85, 49, 12, 53, 44, 91, 79, 23, 3, 19
Parent 2: 23, 95, 1, 67, 35, 81, 7, 17, 51, 102, 56, 39, 72, 3, 54, 37, 13, 86, 47, 76

Child 1: 2, 18, 36, 27, 11, 13, 6, 17, 51, 102, 56, 39, 72, 3, 54, 37, 13, 86, 47, 76
Child 2: 23, 95, 1, 67, 35, 81, 7, 43, 64, 12, 85, 49, 12, 53, 44, 91, 79, 23, 3, 19

1. Individual 25 289.5s
2. Individual 17 256.7s
3. Individual 91 197.2s

……
Using GAs in GA-Prof

Individual 1
Individual 2
Individual 3

1. Individual 25 289.5s
2. Individual 17 256.7s
3. Individual 91 197.2s
......

Parent: 2, 18, 36, 27, 11, 13, 6, 43, 64, 12, 85, 49, 12, 53, 44, 91, 79, 23, 3, 19

Child: 2, 18, 36, 27, 11, 13, 6, 43, 64, 73, 85, 49, 12, 53, 44, 91, 79, 23, 3, 19

crossover
mutation
Using GAs in GA-Prof

Individual 1
Individual 2
Individual 3

1. Individual 25 289.5s
2. Individual 17 256.7s
3. Individual 91 197.2s
......

crossover
mutation

Individual 1'
Individual 2'
Individual 3'
......
Using GAs in GA-Prof

Independent variables:
Crossover rate, mutation rate, number of individuals per generation
Identifying input-sensitive bottlenecks

Elapsed Execution Time

<table>
<thead>
<tr>
<th>Traces</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>137.92</td>
</tr>
<tr>
<td></td>
<td>233.2</td>
</tr>
<tr>
<td></td>
<td>128.48</td>
</tr>
<tr>
<td></td>
<td>79.6</td>
</tr>
<tr>
<td></td>
<td>201.8</td>
</tr>
</tbody>
</table>
Identifying input-sensitive bottlenecks

<table>
<thead>
<tr>
<th>Elapsed Execution Time</th>
<th>Traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>137.92</td>
<td>Bad</td>
</tr>
<tr>
<td>233.2</td>
<td>Good</td>
</tr>
<tr>
<td>128.48</td>
<td>Bad</td>
</tr>
<tr>
<td>79.6</td>
<td>Bad</td>
</tr>
<tr>
<td>201.8</td>
<td>Good</td>
</tr>
</tbody>
</table>
Identifying input-sensitive bottlenecks

- Method 1
- Method 2
- Method 3
- Method 4
- Method 5

Elapsed Execution Time

Traces

Good

Bad
Identifying input-sensitive bottlenecks

- Method 1
- Method 2
- Method 3
- Method 4
- Method 5

Elapsed Execution Time

Good

Bad

bottleneck

Good

Traces

39
Independent Component Analysis (ICA)

Independent Component Analysis: Algorithms and Applications
Aapo Hyvärinen and Erkki Oja
Neural Networks Research Centre
Helsinki University of Technology
P.O. Box 5400, FIN-02015 HUT, Finland
Neural Networks, 13(4-5):411-430, 2000

Abstract
A fundamental problem in neural network research, as well as in many other disciplines, is finding a suitable representation of multivariate data, i.e., random vectors. For reasons of computational and conceptual simplicity, the representation is often sought as a linear transformation of the original data. In other words, each component is designed to help isolate and relate to specific concepts. Techniques like Singular Value Decomposition are used to help solve this problem.

Automated Concept Location Using Independent Component Analysis
Scott Grant, James R. Cordy, David Skillicorn
School of Computing, Queen’s University
Kingston, Ontario, Canada
(scott, cordy, skill)@cs.queensu.ca

Independent Component Analysis (ICA) [3, 5] is a blind signal separation technique that separates a set of input signals into statistically independent components. It operates in a similar way to Singular Value Decomposition, which is
Independent Component Analysis (ICA)
Independent Component Analysis (ICA)
Independent Component Analysis (ICA)

Feature 1

Feature 2

ICA

Trace 1

Trace 2
Independent Component Analysis (ICA)

\[
\begin{bmatrix}
\text{traces} \\
\text{methods}
\end{bmatrix}
\times
\begin{bmatrix}
\text{features} \\
\text{traces}
\end{bmatrix}
= 
\begin{bmatrix}
\text{features} \\
\text{methods}
\end{bmatrix}
\]

\[X: p \times m \quad A: p \times k \quad S: k \times m\]

Good traces \[\Rightarrow [S_{\text{Good}}]\]  
Bad traces \[\Rightarrow [S_{\text{Bad}}]\]
Contrast Mining

For each method

\[ D = \sum (S_{Good} - S_{Bad}) \]
Inputs → App → Profiler

Collect execution traces

MTDENT_|_http-bio-8080-exec-7_|_131011218033882_|_fi/hut/soberit/agilefant/model/Story_|_getStoryAccesses()Ljava/util/Set;_||_

MTDRET_|_http-bio-8080-exec-7_|_131011218036560_|_fi/hut/soberit/agilefant/model/Story_|_getStoryAccesses()Ljava/util/Set;_||_

MTDENT_|_http-bio-8080-exec-7_|_131011218074491_|_fi/hut/soberit/agilefant/model/User_|_getId()I_||_

MTDRET_|_http-bio-8080-exec-7_|_131011218076276_|_fi/hut/soberit/agilefant/model/User_|_getId()I_||_
1. Seq of inputs 25  289.5s  
2. Seq of inputs 17  256.7s  
3. Seq of inputs 91  197.2s  
......

crossover

mutation

Seq of inputs 1'  
Seq of inputs 2'  
Seq of inputs 3'
Inputs → App → Profiler → Execution Trace Analyzer → Trace Statistics → GA Analyzer

Method Statistics → ICA

Good Traces ← ← ← ← Bad Traces

Good traces → $[S_{\text{Good}}]$  Bad traces → $[S_{\text{Bad}}]$
For each method

$$D = \sum (S_{Good} - S_{Bad})$$

1. Agilefant.model.WidgetCollection.getName() 11.783
2. Agilefant.db.hibernate.UserTypeFilter.deepCopy() 10.662
3. Agilefant.model.Team.setId() 8.112
5. ......
Research Questions (RQs)

• RQ₁ - How effective is GA-Prof in finding inputs leading to bottlenecks

• RQ₂ - How effective is GA-Prof in identifying bottlenecks

• RQ₃ - Is GA-Prof more effective than FOREPOST in identifying bottlenecks
Research Question 1

How effective is GA-Prof in finding inputs leading to bottlenecks

GA-Prof vs. Random
Research Question 1

How effective is GA-Prof in finding inputs leading to bottlenecks

GA-Prof vs. Random

$H_0$: There is no statistical difference between GA-Prof and Random
Research Question 2

How effective is GA-Prof in identifying bottlenecks

Inject nine artificial performance bottlenecks
Research Question 3

Is GA-Prof more effective than FOREPOST in identifying bottlenecks

GA-Prof vs. FOREPOST
Experimental Design

JPetStore

Agilefant

Dell DVD Store

<table>
<thead>
<tr>
<th>DVD Store</th>
<th>Selected items: specify quantity desired and click Update; click Purchase when finished</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>Quantity: 12.00</td>
</tr>
<tr>
<td></td>
<td>Subtotal: 148.89</td>
</tr>
<tr>
<td>Thank You for Visiting the DVD Store!</td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2009 Dell
Experimental Design

5 users

50 URLs per user

30 times to repeat
RQ$_1$ – Finding Bottleneck-Specific Inputs

### Generations

<table>
<thead>
<tr>
<th>Generations</th>
<th>Elapsed Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>18</td>
<td>22</td>
</tr>
<tr>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>22</td>
<td>26</td>
</tr>
<tr>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td>26</td>
<td>30</td>
</tr>
<tr>
<td>28</td>
<td>32</td>
</tr>
</tbody>
</table>

**JPetStore**
RQ₁ – Finding Bottleneck-Specific Inputs

Dell DVD Store

Elapsed Execution Time

Generations
RQ₁ – Finding Bottleneck-Specific Inputs
RQ₁ – Finding Bottleneck-Specific Inputs

Elapsed Execution Time

JPetStore

- Random
- GA-Prof

Dell DVD Store

- Random
- GA-Prof

Agilefant

- Random
- GA-Prof
RQ$_1$ – Finding Bottleneck-Specific Inputs

Null hypothesis is Rejected

$p = 1.5e - 21$

$p = 2.9e - 30$

$p = 6.4e - 17$
RQ₂ – Finding Injected Bottlenecks

Injected bottleneck:
Jpetstore.domain.Product.getName()
RQ$_2$ – Finding Injected Bottlenecks

Injected bottleneck: Jpetstore.domain.Product.getName()
RQ$_2$ – Finding Injected Bottlenecks

Rankings of Nine Injected bottlenecks  X axis – rankings, Y axis – number of generations
Can GA-Prof capture more injected bottlenecks?

<table>
<thead>
<tr>
<th></th>
<th>JPetStore</th>
<th>Dell DVD Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA-Prof</td>
<td>2.2</td>
<td>2.6</td>
</tr>
<tr>
<td>FOREPOST</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

RQ₃ – GA-Prof vs. FOREPOST
RQ₃ – GA-Prof vs. FOREPOST

Can GA-Prof capture more injected bottlenecks?

<table>
<thead>
<tr>
<th></th>
<th>JPetStore</th>
<th>Dell DVD Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of captured bottlenecks</td>
<td>2.2</td>
<td>2.6</td>
</tr>
<tr>
<td>FOREPOST</td>
<td>5.6</td>
<td>4.6</td>
</tr>
<tr>
<td>GA-Prof</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

68
RQ3 – GA-Prof vs. FOREPOST

Can GA-Prof rank injected performance bottlenecks higher?

<table>
<thead>
<tr>
<th>Service</th>
<th>Average Rankings</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPetStore</td>
<td>145.98</td>
</tr>
<tr>
<td>Dell DVD Store</td>
<td>14.8</td>
</tr>
</tbody>
</table>

FOREPOST
RQ₃ – GA-Prof vs. FOREPOST

Can GA-Prof rank injected performance bottlenecks higher?

Average rankings

```
<table>
<thead>
<tr>
<th></th>
<th>FOREPOST</th>
<th>GA-Prof</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPetStore</td>
<td>145.98</td>
<td>13.78</td>
</tr>
<tr>
<td>Dell DVD Store</td>
<td>14.8</td>
<td>10.94</td>
</tr>
</tbody>
</table>
```


Automating Performance Bottleneck Detection using Search-Based Application Profiling

Du Shen, Qi Luo, Denys Poshyvanyk
Department of Computer Science
College of William and Mary
Williamsburg, Virginia 23185, USA
dshen,qluo,densys@cs.wm.edu

Mark Grechanik
Department of Computer Science
University of Illinois at Chicago
Chicago, Illinois 60607, USA
drmark@uic.edu

ABSTRACT
Application profiling is an important performance analysis technique, when an application under test is analyzed dynamically to determine its space and time complexities and the usage of its instructions. A big and important challenge is to profile nontrivial web applications with large numbers of combinations of their input parameter values. Identifying and understanding particular subset-

1. INTRODUCTION

Improving performance of software applications is one of the most important tasks in software evolution and maintenance [16]. Software engineers make performance enhancements routinely during perfective maintenance [55] when they use exploratory random performance testing [25, 11] to identify methods that lead to performance bottlenecks (or hot spots), which are phenomena when...
Input-sensitive profiling

**Intellectually intensive and time-consuming**

1. Agilefant.model.WidgetCollection.getName()  273.2s
2. Agilefant.db.hibernate.UserTypeFilter.deepCopy()  213.5s
3. Agilefant.model.Team.setld()  192.3s
4. Agilefant.model.Backlog.setChildren()  123.9s
5. .......
Input-sensitive profiling

Intellectually intensive and time-consuming

1. Agilefant.model.WidgetCollection.getName() 273.2s
2. Agilefant.db.hibernate.UserTypeFilter.deepCopy() 213.5s
3. Agilefant.model.Team.setId() 192.3s
4. Agilefant.model.Backlog.setChildren() 123.9s
5. .......

Genetic Algorithm-driven Profiler (GA-Prof)

Ranked list:
1. Agilefant.db.hibernate.UserTypeFilter.deepCopy()
2. Agilefant.model.WidgetCollection.getName()
3. Agilefant.model.Team.setId()
Input-sensitive profiling

Intellectually intensive and time-consuming

1. Agileant.model.WidgetCollection.getName()  273.2s
2. Agileant.db.hibernate.UserTypeFilter.deepCopy()  213.5s
3. Agileant.model.Team.setid()  192.3s
4. Agileant.model.Backlog.setChildren()  123.9s
5. .......

Genetic Algorithm-driven Profiler (GA-Prof)

Software system
Profiler
GA Analyzer
Contrast
Mining

Ranked list:
1. Agileant.db.hibernate.UserTypeFilter.deepCopy()
2. Agileant.model.WidgetCollection.setName()
3. Agileant.model.Backlog.setChildren()
4. Agileant.model.Team.setid()
5. .......

Research Questions (RQs)

• RQ₁ - How effective is GA-Prof in finding inputs leading to bottlenecks
• RQ₂ - How effective is GA-Prof in identifying bottlenecks
• RQ₃ - Is GA-Prof more effective than FOREPOST in identifying bottlenecks
Input-sensitive profiling

Intellectually intensive and time-consuming

1. Agilefant.model.WidgetCollection.getName() 273.2s
2. Agilefant.db.hibernate.UserTypeFilter.deepCopy() 213.5s
3. Agilefant.model.Team.setId() 192.3s
4. Agilefant.model.Backlog.setChildren() 123.9s
5. ... 

Genetic Algorithm-driven Profiler (GA-Prof)

Software system → Profiler → GA Analyzer → Contrast Mining → Ranked list:
1. Agilefant.db.hibernate.UserTypeFilter.deepCopy()
2. Agilefant.model.WidgetCollection.setName()
4. Agilefant.model.Team.setId()
5. ... 

Research Questions (RQs)

- RQ1 - How effective is GA-Prof in finding inputs leading to bottlenecks
- RQ2 - How effective is GA-Prof in identifying bottlenecks
- RQ3 - Is GA-Prof more effective than FOREPOST in identifying bottlenecks

Experimental Results

- GA-Prof is effective in finding inputs leading to performance bottlenecks
- GA-Prof is effective in identifying bottlenecks
- As compared to FOREPOST, GA-Prof can capture more injected bottlenecks and rank them higher
Additional Slides for Questions
GAs – Independent Variables

- Crossover rate – 0.3
- Mutation rate – 0.1
- Number of individuals per generation – 30
- Termination Criterion
  - Maximum limit for the number of generations – 30
  - Average fitness value of every individual in one generation
Inject Artificial Performance Bottlenecks

• Run applications without artificial bottlenecks
• Obtain a ranked list of methods
• Randomly inject nine artificial bottlenecks (each one as a same delay)
• Delays are chosen experimentally