Automating Performance Bottleneck Detection using Search-Based Application Profiling

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

College of William and Mary

*University of Illinois at Chicago

ISSTA 2015

Baltimore, MD, U.S.

ing]: Metrics—performance measures

General Terms Algorith

Keyw ...ptotic analysis, dynamic
progr... instrumentation.

Coppa et al. PLDI'12

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

Package	<Base Time (seconds)	Average Base Time (seconds)	Cumulative Time (seconds)	Calls
com.ibm.team.collaboration.internal.core.service	249.406940	0.313325	253.997890	796
DefaultCollaborationService$CollaborationServiceJobQueue	244.485136	12.224257	244.485136	20
dequeue() com.ibm.team.collaboration.core.service.CollaborationServiceJob	244.484838	40.747473	244.484838	6
enqueue(com.ibm.team.collaboration.core.service.CollaborationServiceJob) vo	0.000133	0.000027	0.000133	5
isDisposed() boolean	0.000118	0.000017	0.000118	7
DefaultCollaborationService$CollaborationServiceJobQueue()	0.000036	0.000036	0.000036	1
dispose() void	0.000011	0.000011	0.000011	[illegible]

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

Inputs

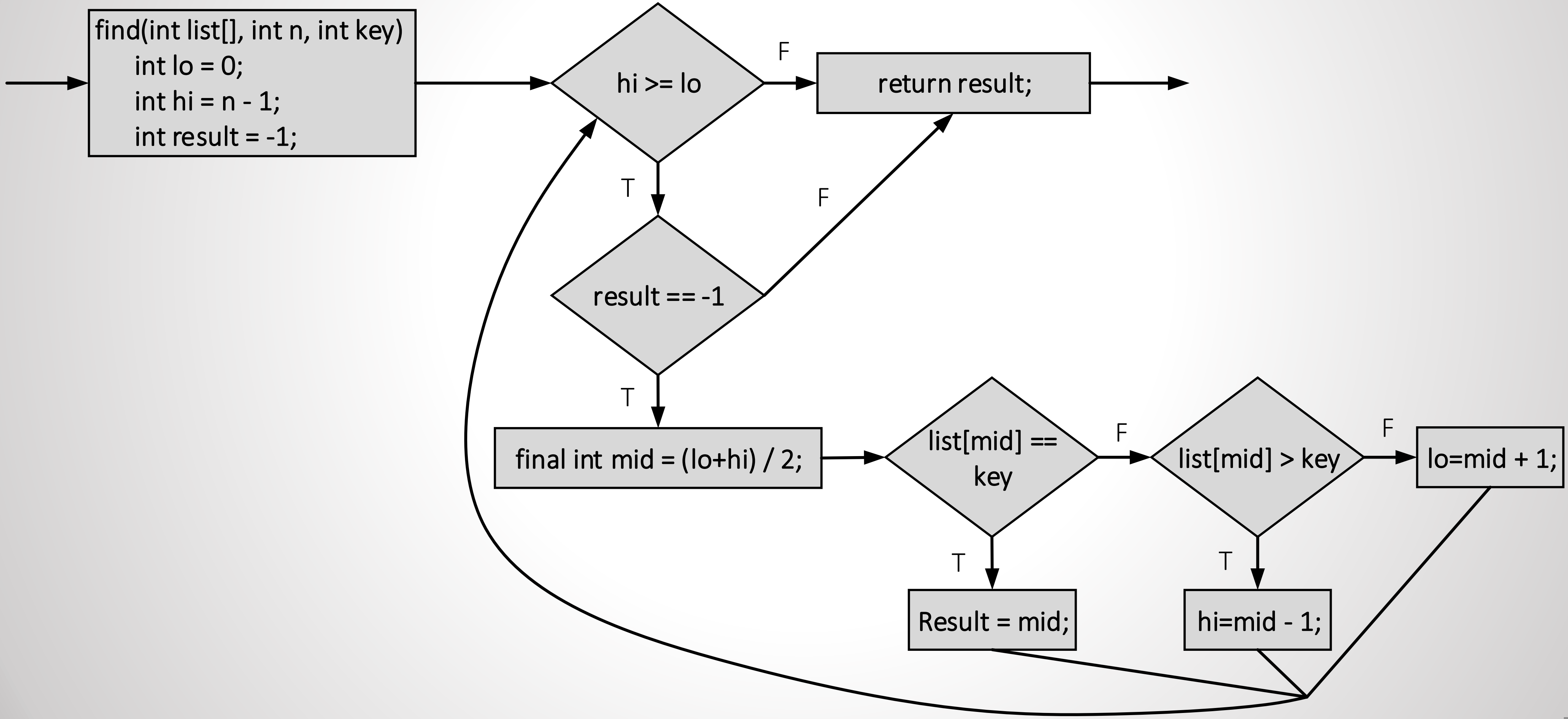
Application

CollectionSpy
JProfiler
ADB
JProbe
Emmagee
TPTP Probekit
OKTECH
JRockit
DDMS
TraceView
Eclipse
Netbeans

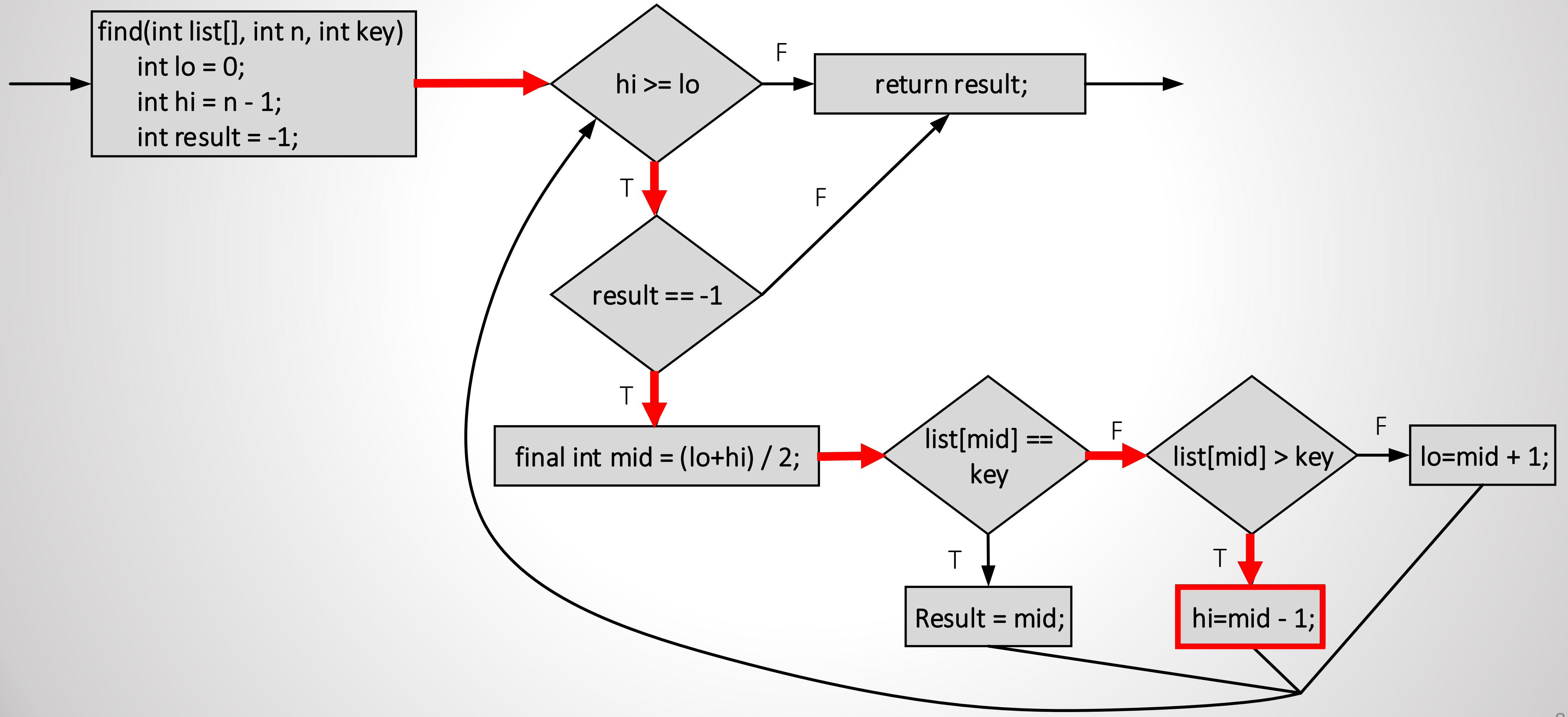
Its success depends on the chosen set of input values

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. ……	

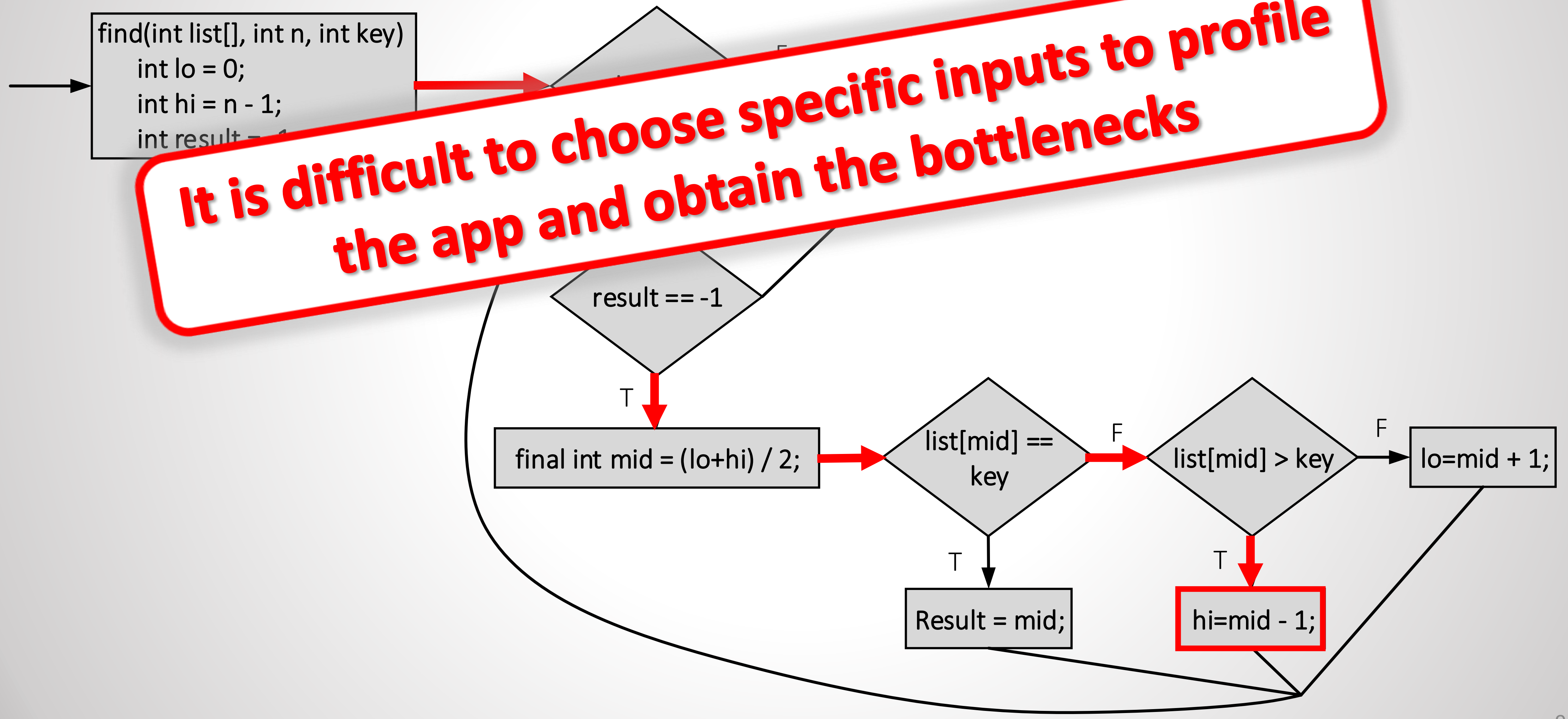
Standard application profiling



Standard application profiling

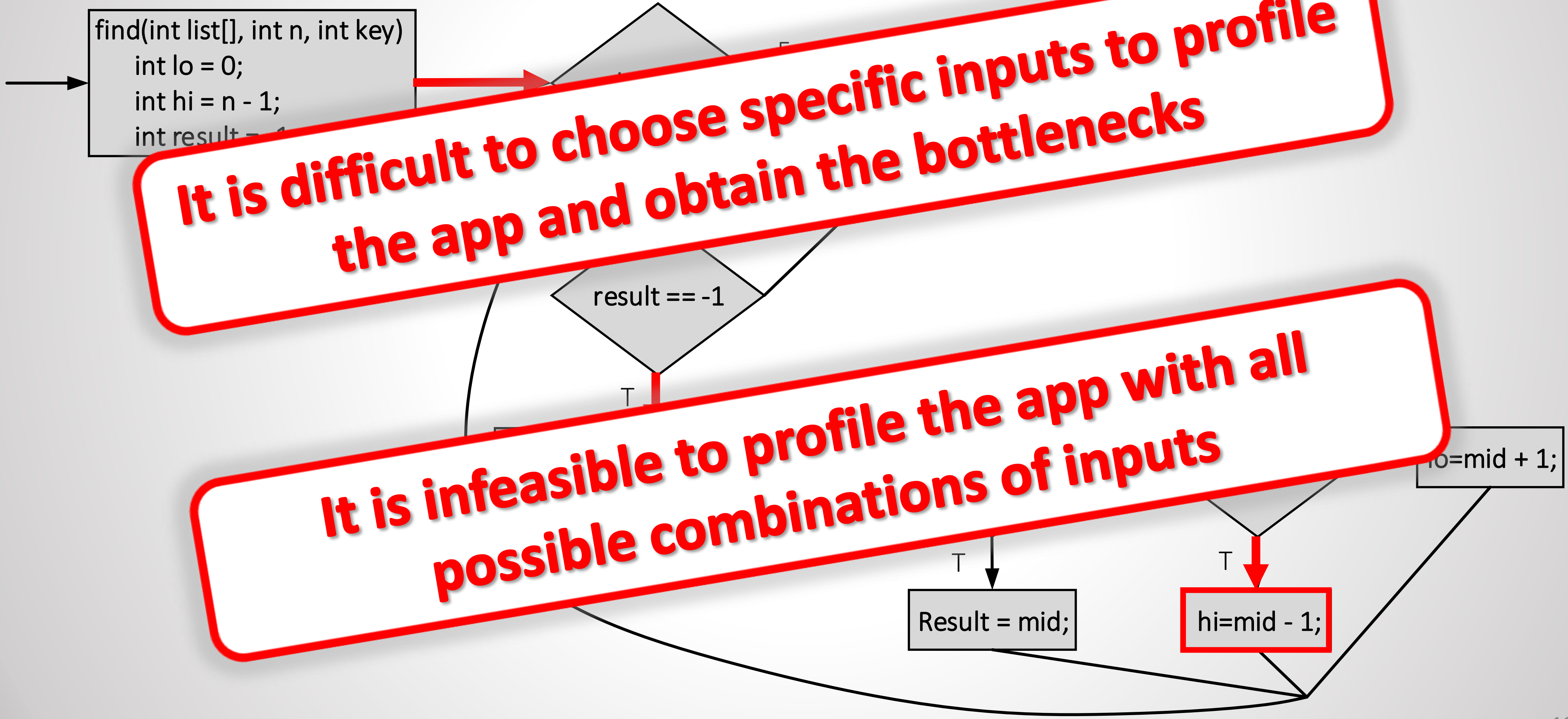


Standard application profiling



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 pro-

the cost of a given run, 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 approx-

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

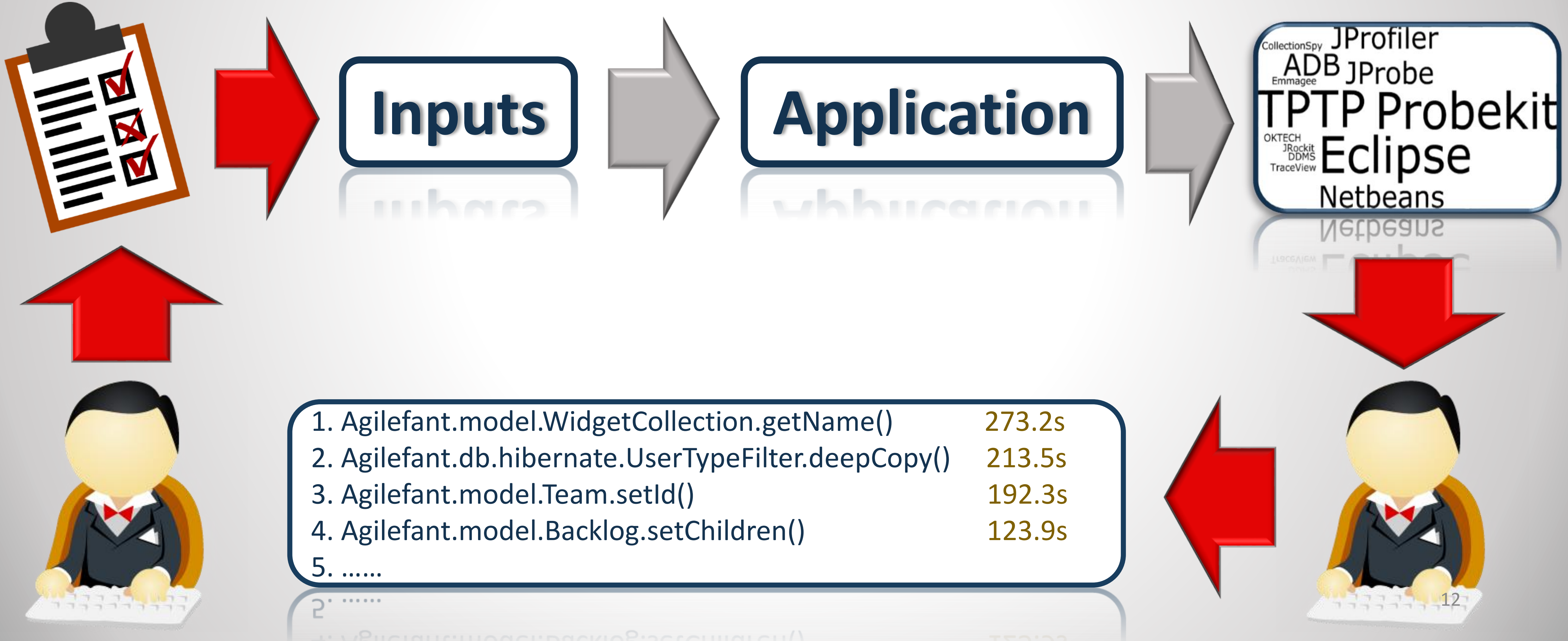
Irene Finocchi
Dept. of Computer Science
Sapienza University of Rome
finocchi@di.uniroma1.it

Abstract

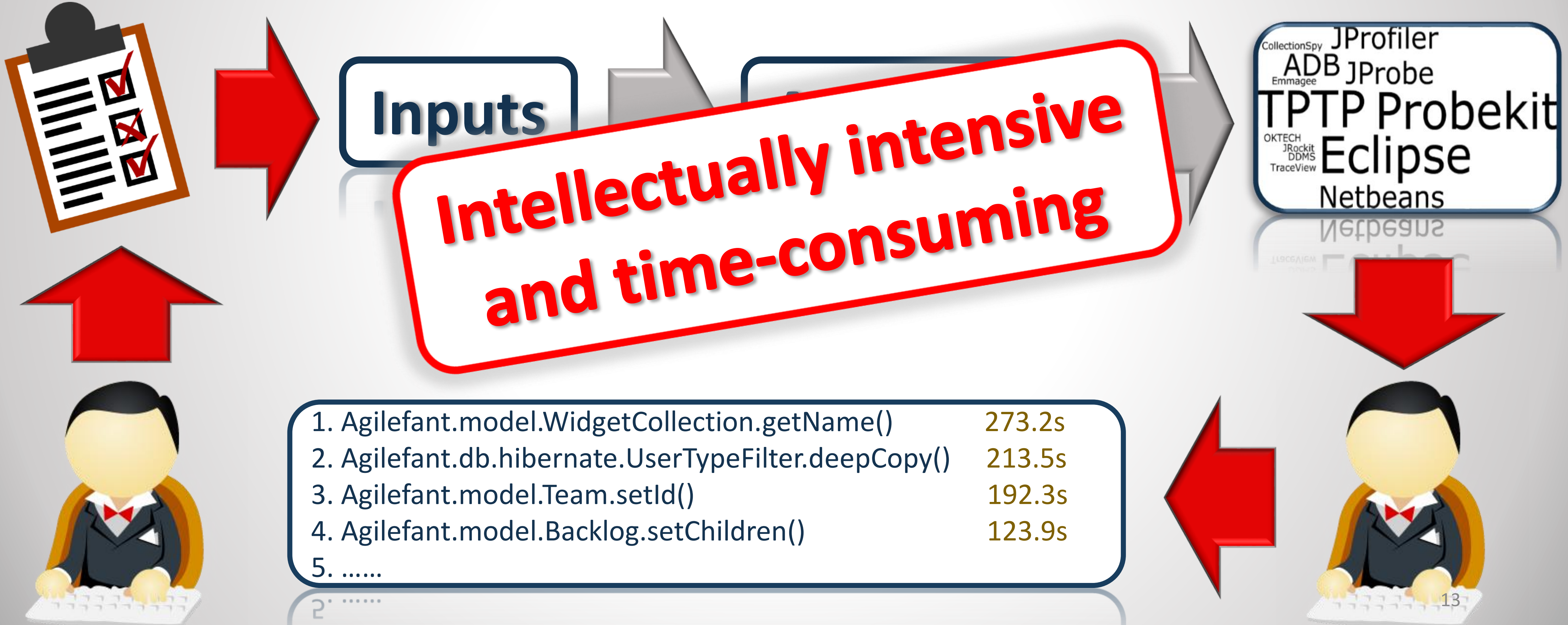
In this paper we present a profiling methodology and toolkit for helping developers discover hidden asymptotic inefficiencies in the code. From one or more runs of a program, our profiler automatically measures how the performance of individual routines scales

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

Input-sensitive profiling



Input-sensitive profiling



Input-sensitive profiling

```
Input  x, y, z, u

v = A.m (x, y)
if (v > z) {
    C.h (B.m (v))
}
else {
    D.h (B.m (v))
}
```

Input-sensitive profiling

```
Input  x, y, z, u

v = A.m (x, y)
if (v > z) {
    C.h (B.m (v))
}
else {
    D.h (B.m (v))
}
```

unimportant input value

Input-sensitive profiling

```
Input  x, y, z, u

v = A.m (x, y)
if (v > z) {
    C.h (B.m (v))
}
else {
    D.h (B.m (v))
}
```

Construct combinations of input values

Input-sensitive profiling

```
Input  x, y, z, u

v = A.m (x, y)
if (v > z) {
    C.h (B.m (v))
}
else {
    D.h (B.m (v))
}
```

General-purpose methods

Input-sensitive profiling

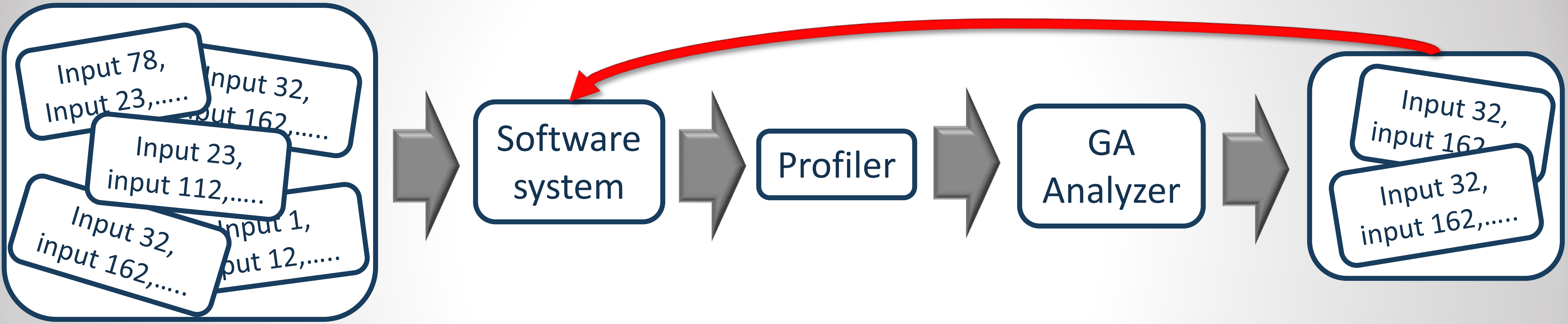
```
Input  x, y, z, u

v = A.m (x, y)
if (v > z) {
    C.h (B.m (v))
}
else {
    D.h (B.m (v))
}
```

Identify the input-sensitive bottlenecks

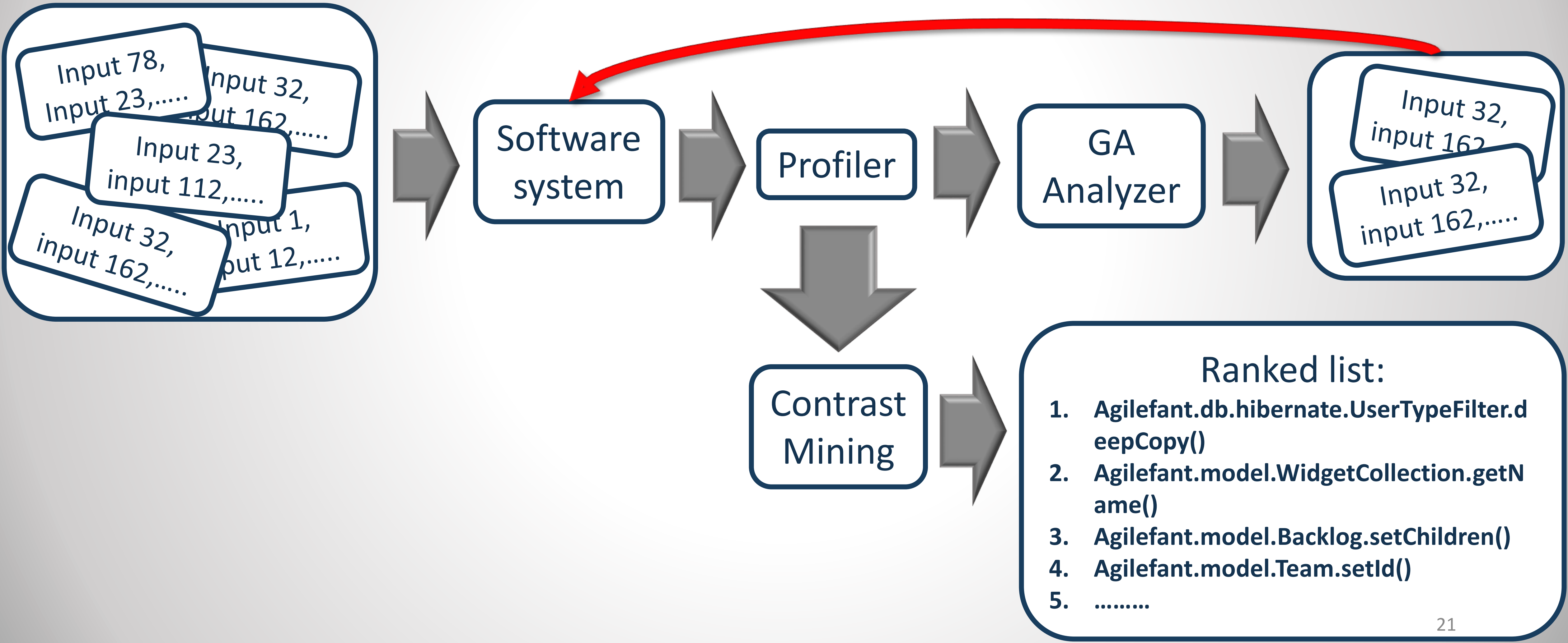
Genetic Algorithm-driven Profiler (GA-Prof)

- Automate input-sensitive profiling
- Explore input parameter space
- Detect performance bottlenecks

Genetic Algorithm-driven Profiler (GA-Prof)



Genetic Algorithm-driven Profiler (GA-Prof)



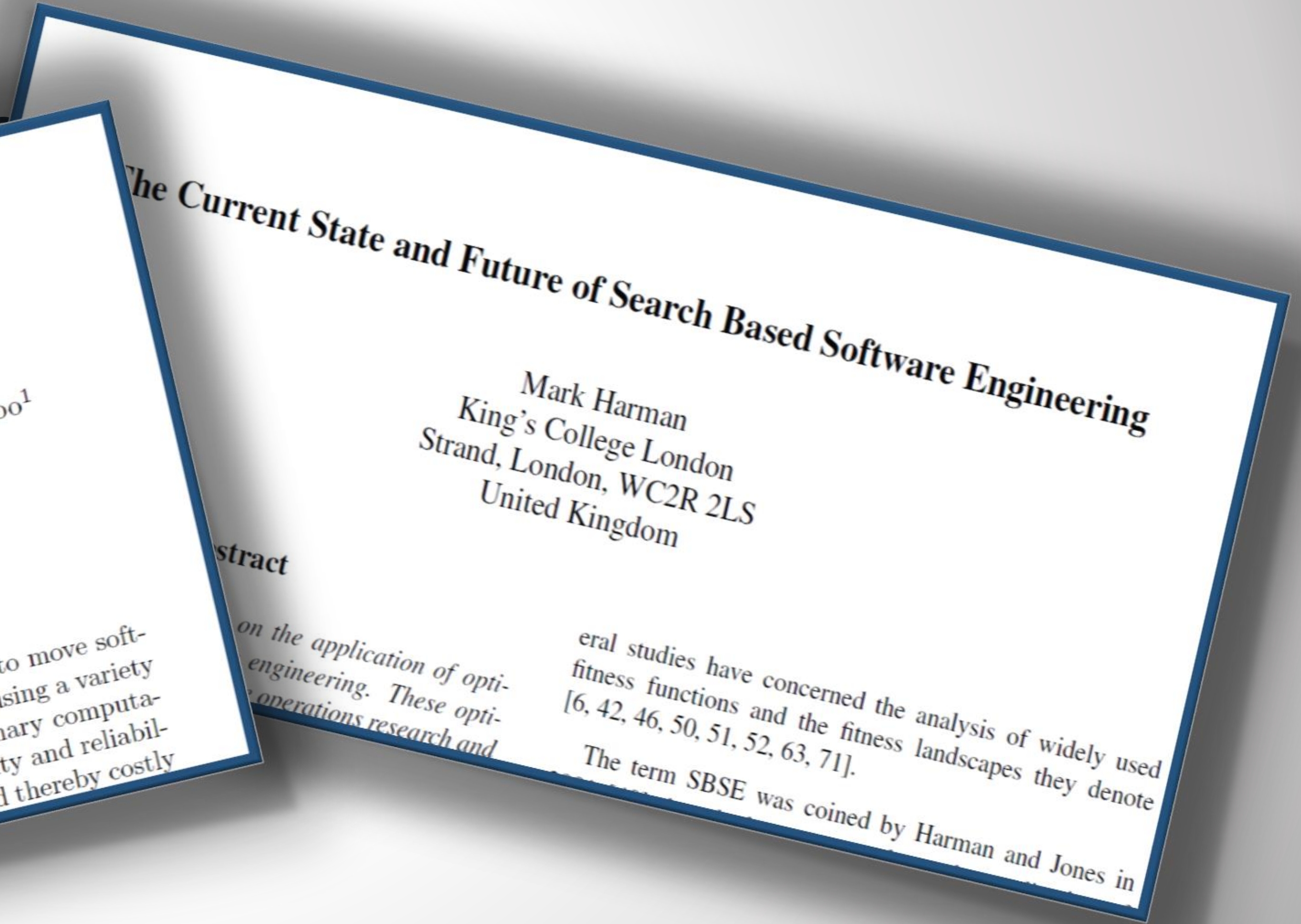
Genetic Algorithms (GAs)

- Simulate the natural selection process
- Generate solutions to optimization problems

Genetic Algorithms (GAs)

- Simulate the natural sel

- Ge



he Current State and Future of Search Based Software Engineering

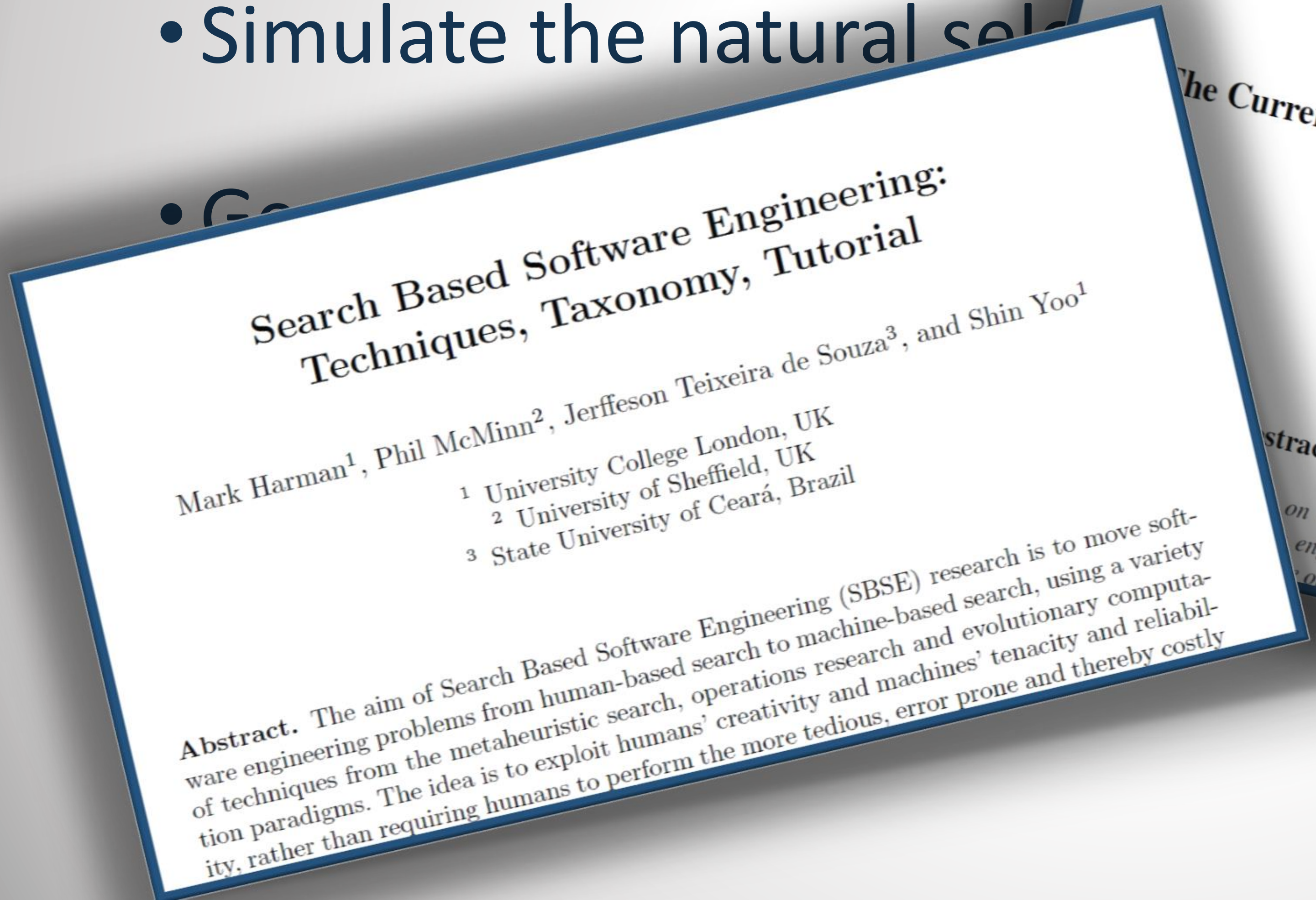
Mark Harman
King's College London
Strand, London, WC2R 2LS
United Kingdom

stract

on the application of opti-
engineering. These opti-
operations research and

eral studies have concerned the analysis of widely used fitness functions and the fitness landscapes they denote [6, 42, 46, 50, 51, 52, 63, 71].

The term SBSE was coined by Harman and Jones in



Search Based Software Engineering: Techniques, Taxonomy, Tutorial

Mark Harman[1], Phil McMinn[2], Jerffeson Teixeira de Souza[3], and Shin Yoo[1]

[1] University College London, UK
[2] University of Sheffield, UK
[3] State University of Ceará, Brazil

Abstract. The aim of Search Based Software Engineering (SBSE) research is to move software engineering problems from human-based search to machine-based search, using a variety of techniques from the metaheuristic search, operations research and evolutionary computation paradigms. The idea is to exploit humans' creativity and machines' tenacity and reliability, rather than requiring humans to perform the more tedious, error prone and thereby costly

Why do we use GAs in GA-Prof

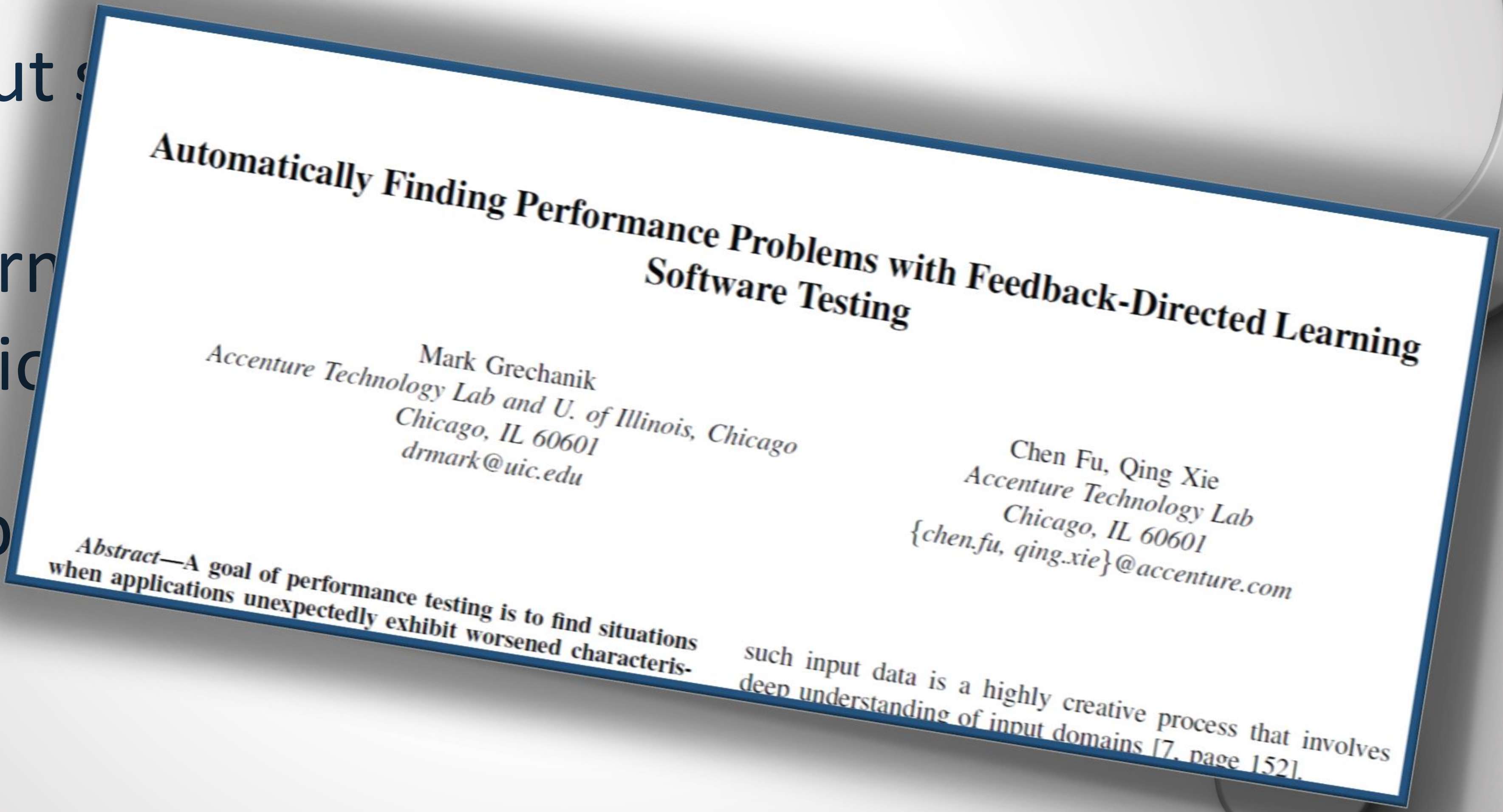
- Large input space
- Can be formulated as a search and optimization problem

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 s
- Can be forr optimizatic
- Perform b solution


Automatically Finding Performance Problems with Feedback-Directed Learning Software Testing

Mark Grechanik
Accenture Technology Lab and U. of Illinois, Chicago
Chicago, IL 60601
drmark@uic.edu

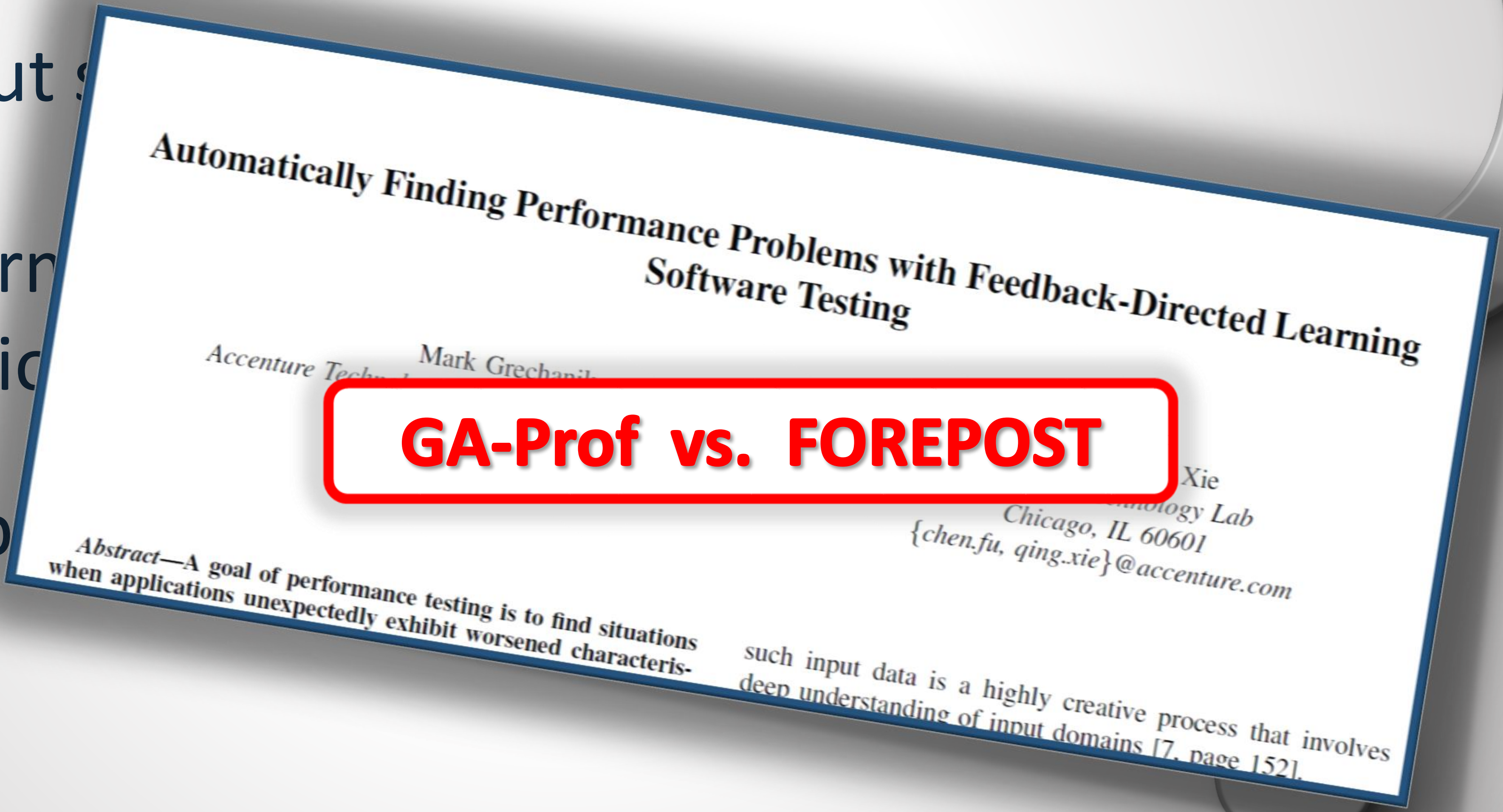
Chen Fu, Qing Xie
Accenture Technology Lab
Chicago, IL 60601
{chen.fu, qing.xie}@accenture.com

Abstract—A goal of performance testing is to find situations when applications unexpectedly exhibit worsened characteris-

such input data is a highly creative process that involves deep understanding of input domains [7, page 152].

Why do we use GAs in GA-Prof

- Large input
- Can be for
  optimizati
- Perform b
  solution



Automatically Finding Performance Problems with Feedback-Directed Learning Software Testing

Chicago, IL 60601
{chen.fu, qing.xie}@accenture.com

Abstract—A goal of performance testing is to find situations when applications unexpectedly exhibit worsened characteris-

such input data is a highly creative process that involves

GA-Prof vs. FOREPOST

GA Component Definitions

Genes:

Input 1: http://localhost:8080/Agilefant/editUser.action
Input 2: http://localhost:8080/Agilefant/editProduct.action?productId=5
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

Individual 1

Individual 2

Individual 3

……

Fitness function:

Individual ⟷ elapsed execution time

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

……

Fitness function:

Individual ⟺ elapsed execution time

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

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

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

……

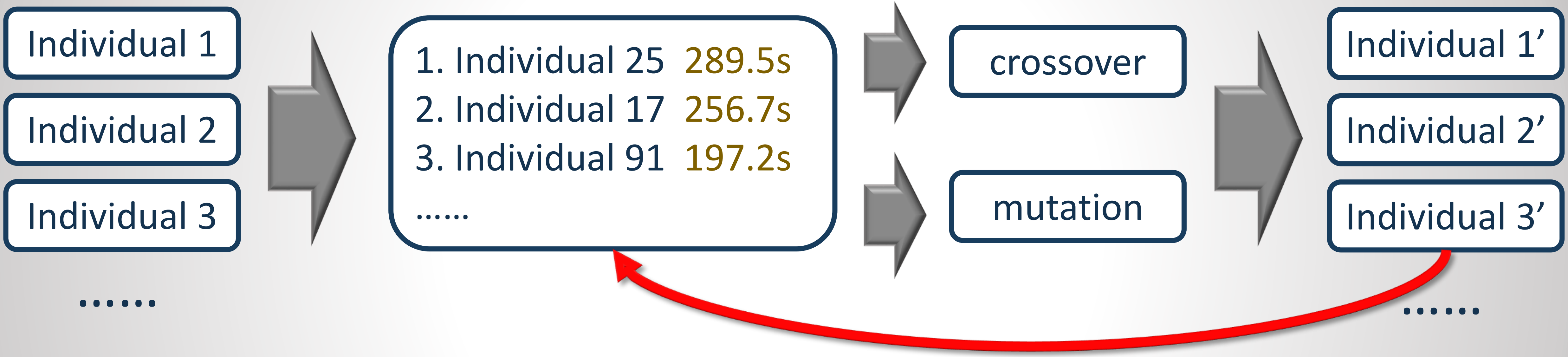
crossover

mutation

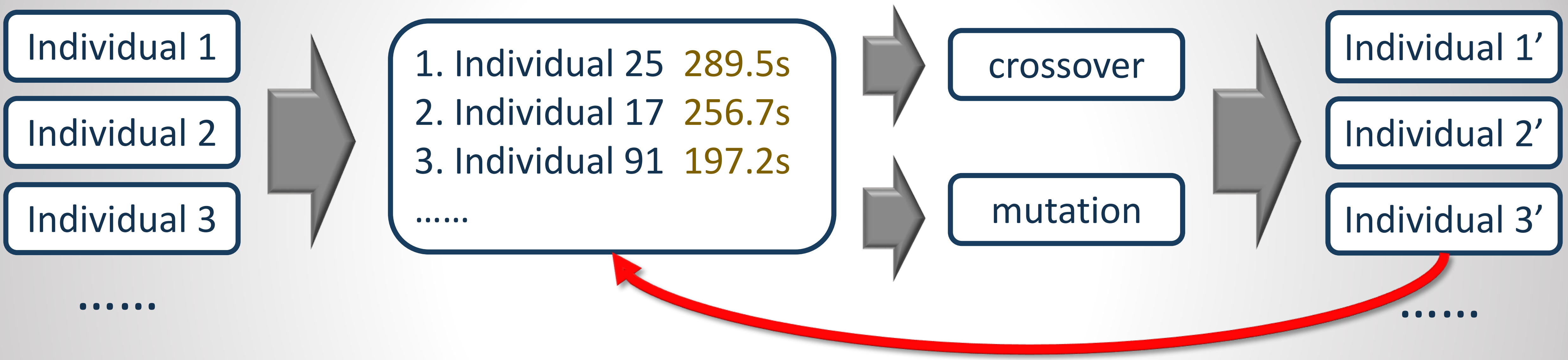
Parent : 2, 18, 36, 27, 11, 13, 6, 43, 64, 12, 85, 49, 12, 53, 44, 91, 79, 23, 3, 19

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

Using GAs in GA-Prof



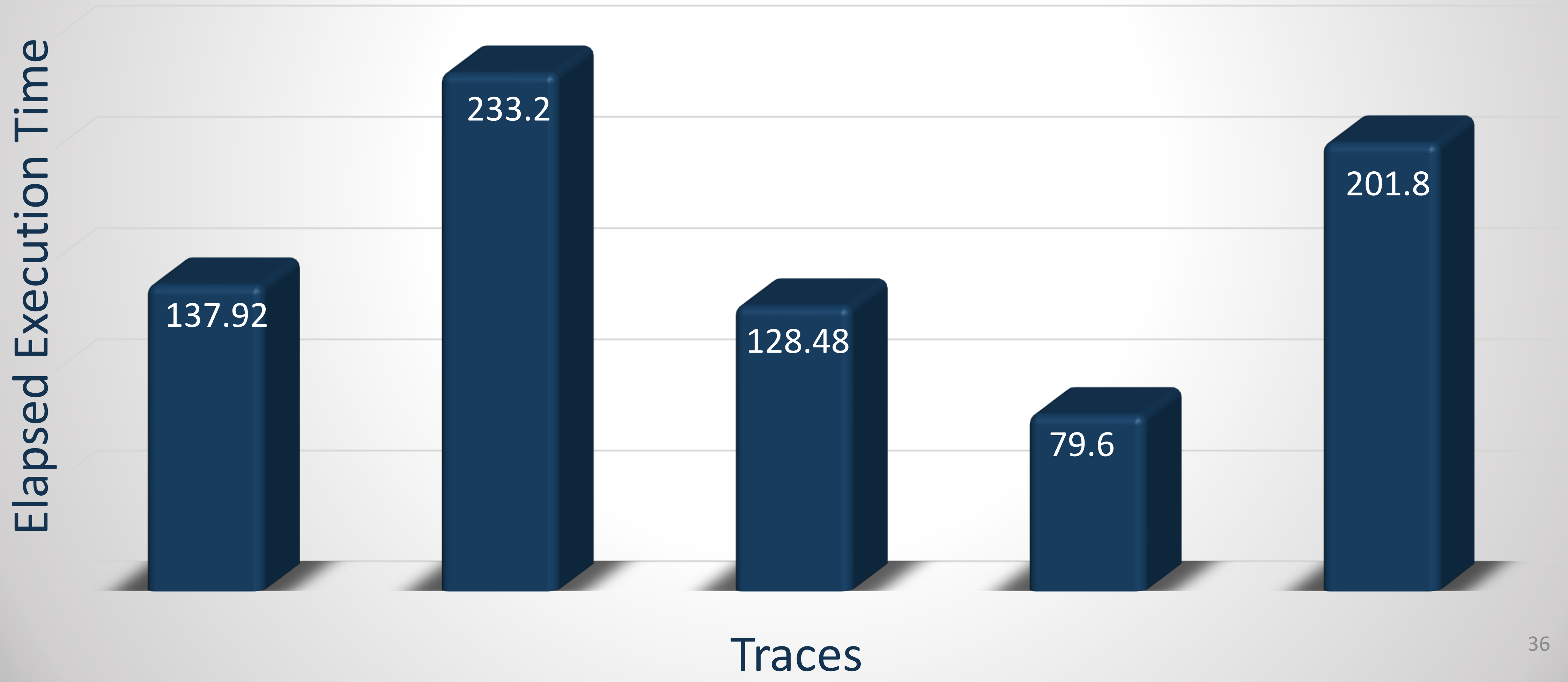
Using GAs in GA-Prof



Independent variables:

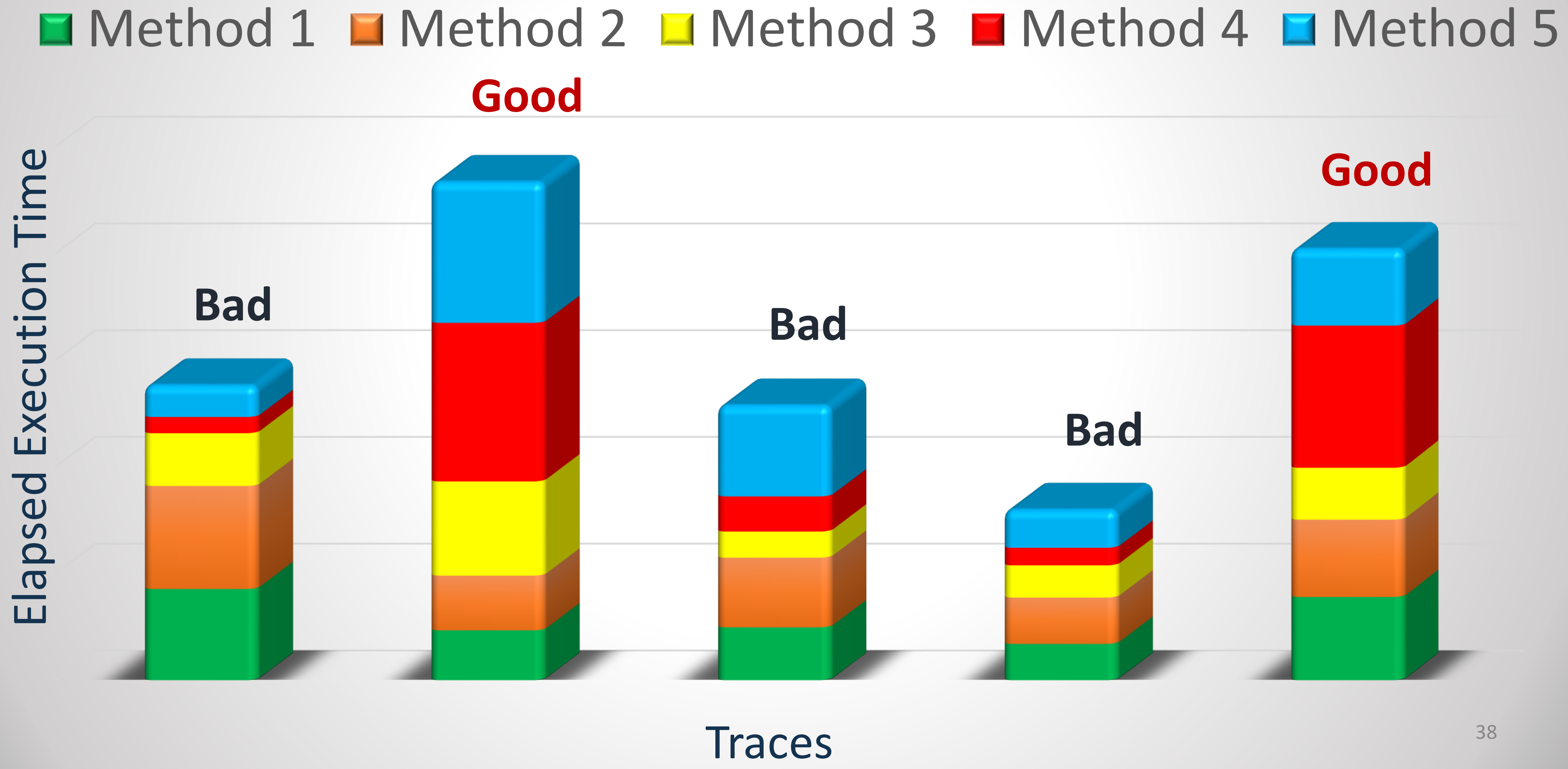
Crossover rate, mutation rate, number of individuals per generation

Identifying input-sensitive bottlenecks

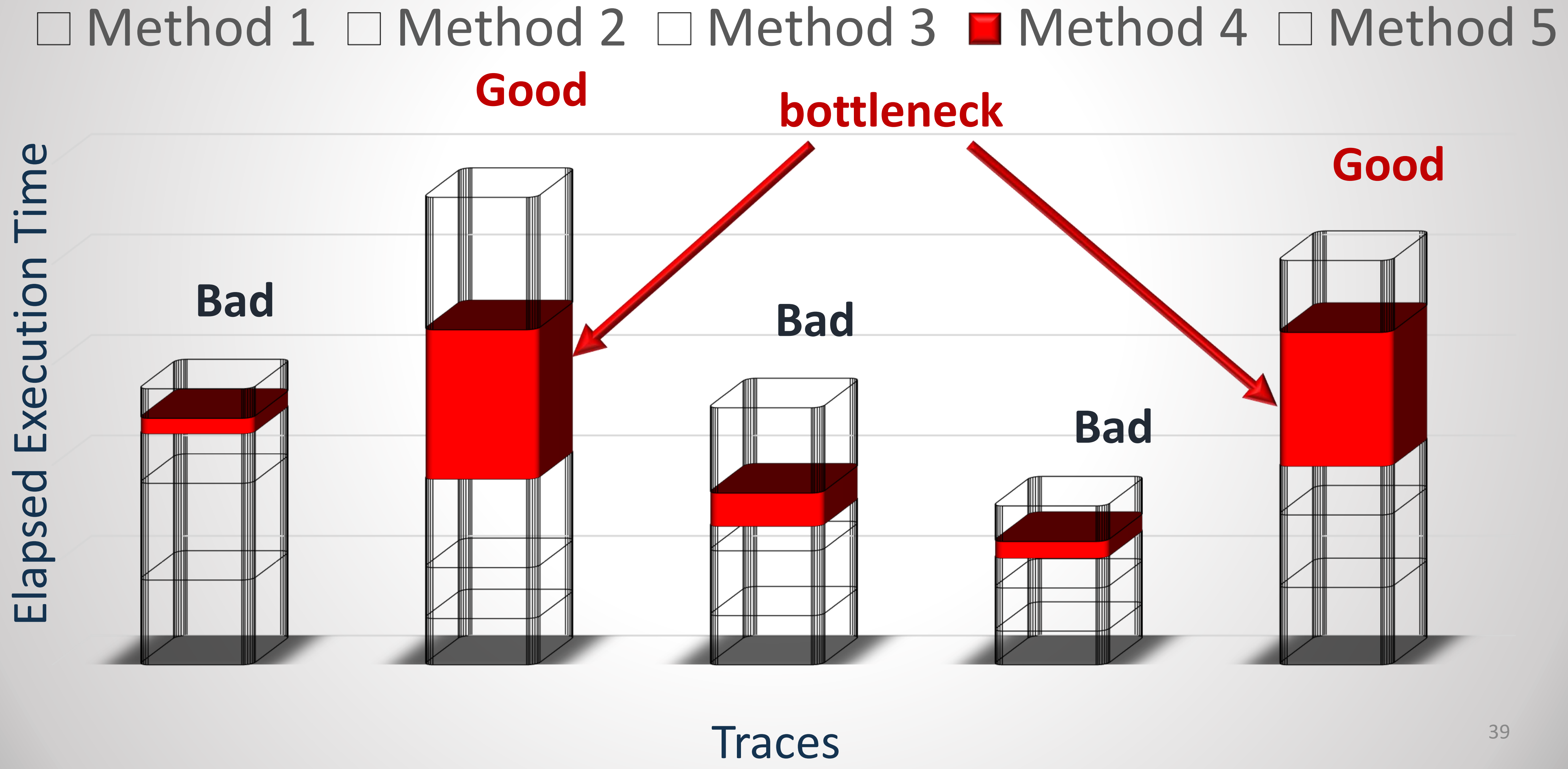


Identifying input-sensitive bottlenecks

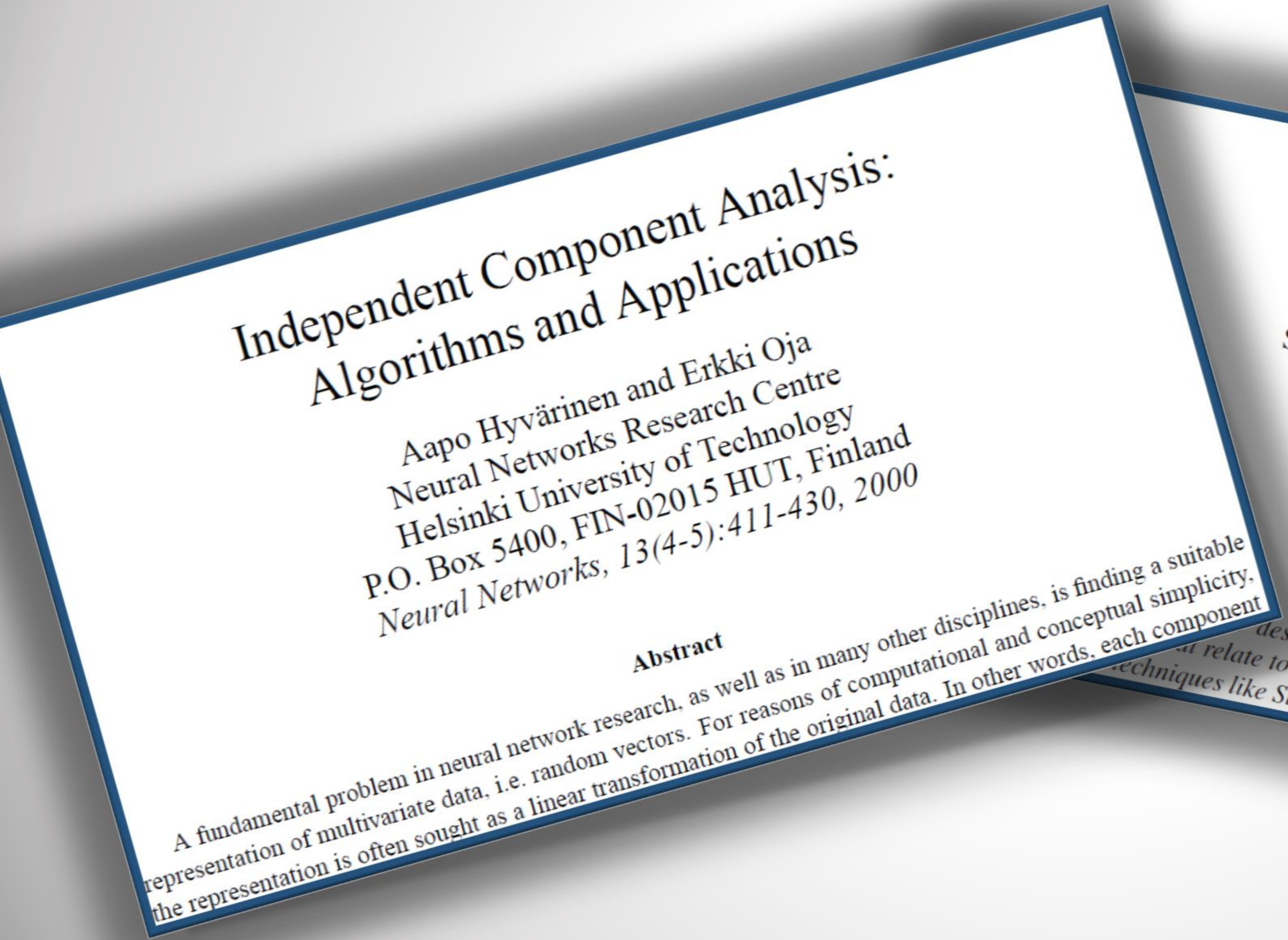
Identifying input-sensitive bottlenecks



Identifying input-sensitive bottlenecks



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


Automated Concept Location Using Independent Component Analysis

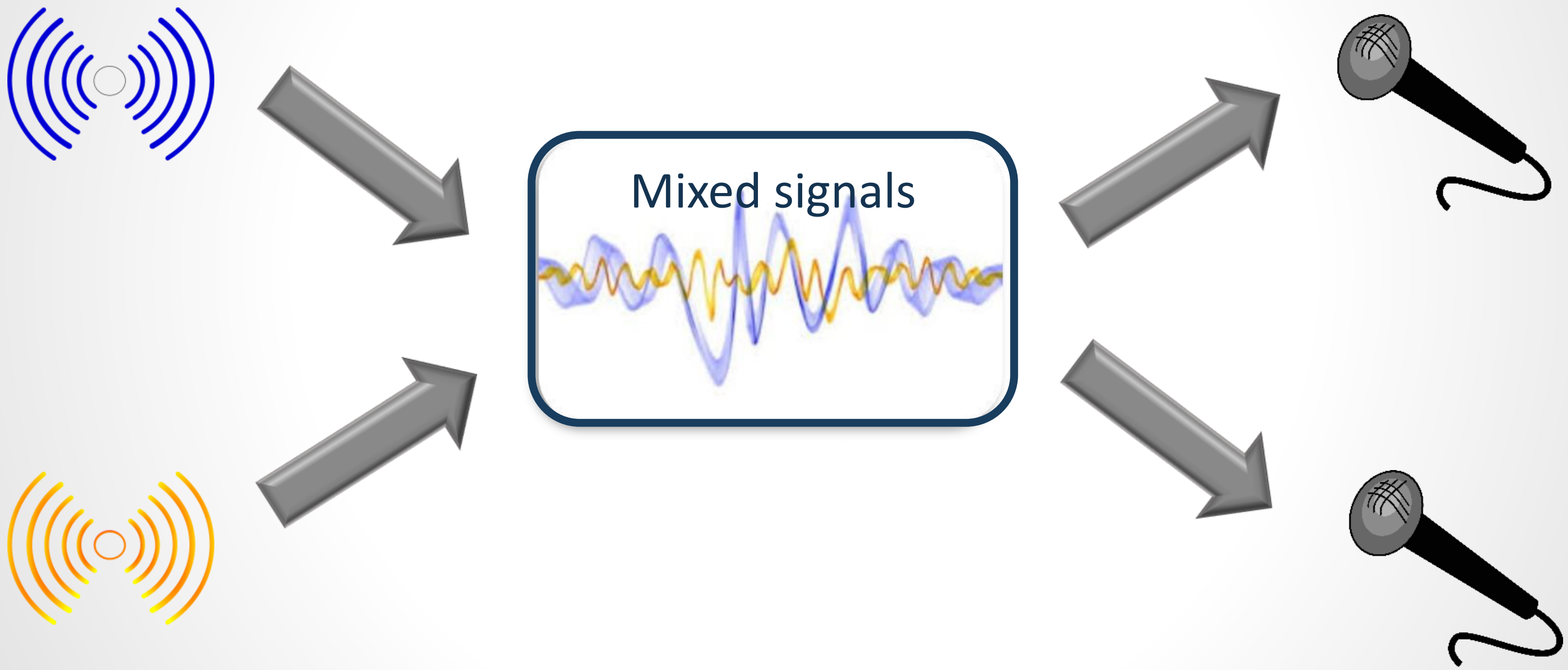
Scott Grant, James R. Cordy, David Skillicorn
School of Computing, Queen's University
Kingston, Ontario, Canada
(scott, cordy, skill)@cs.queensu.ca

designed to help iso-
...l relate to specific concepts.
...echniques like Singular Value De-

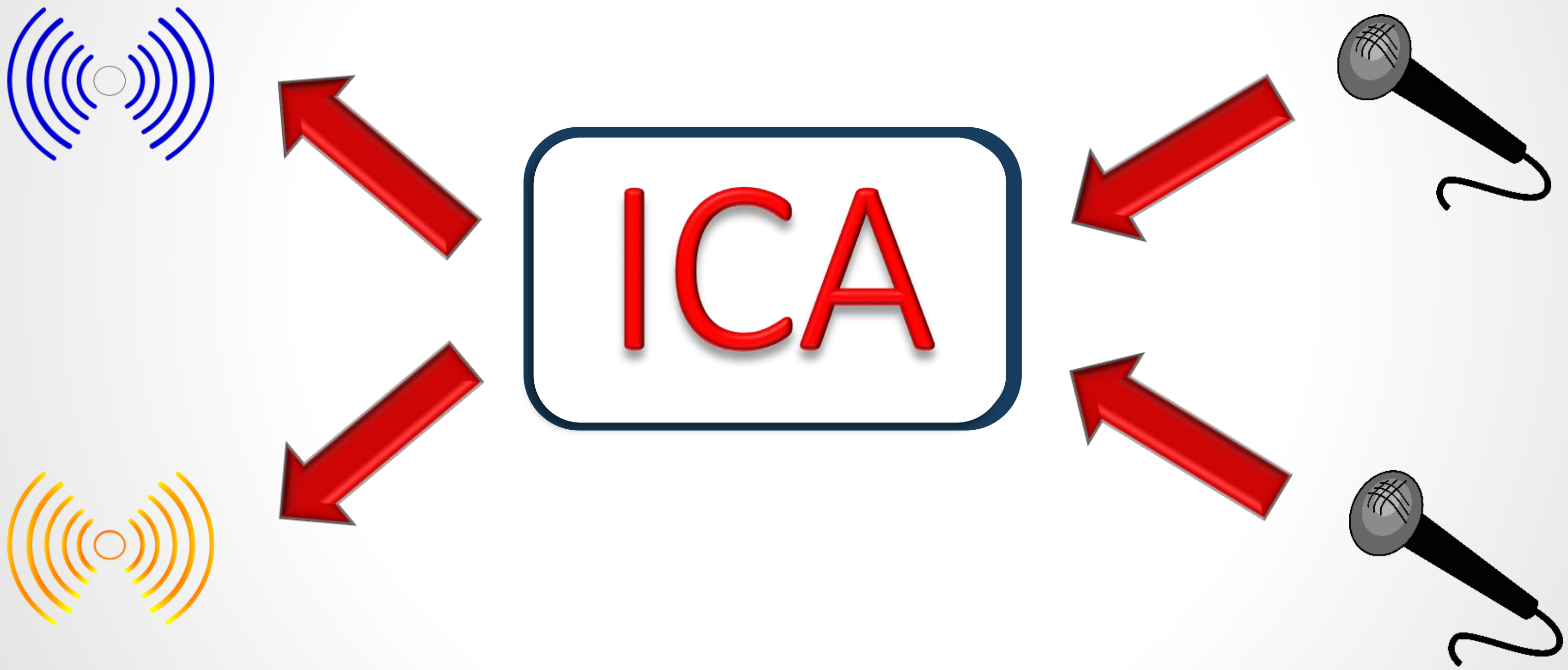
tion problem.

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)

Independent Component Analysis (ICA)

$$\begin{array}{cc} & \text{methods} \\ \text{traces} & \begin{bmatrix} \quad & \quad \\ \quad & \quad \end{bmatrix} \end{array} = \begin{array}{cc} & \text{features} \\ \text{traces} & \begin{bmatrix} \quad \\ \quad \end{bmatrix} \end{array} \times \begin{array}{cc} & \text{methods} \\ \text{features} & \begin{bmatrix} \quad & \quad \\ \quad & \quad \end{bmatrix} \end{array}$$

$\mathbf{X}$: p×m $\quad$ $\mathbf{A}$: p×k $\quad$ $\mathbf{S}$: k×m

Good traces $\Rightarrow$ [S_{Good}] $\qquad$ Bad traces $\Rightarrow$ [S_{Bad}]

Contrast Mining

$$\begin{array}{c} \\ \text{traces} \end{array}\begin{bmatrix} \text{methods} \\ \\ \\ \end{bmatrix} = \begin{array}{c} \\ \text{traces} \end{array}\begin{bmatrix} \text{features} \\ \\ \\ \end{bmatrix} \times \begin{array}{c} \\ \text{features} \end{array}\begin{bmatrix} \text{methods} \\ \\ \end{bmatrix}$$

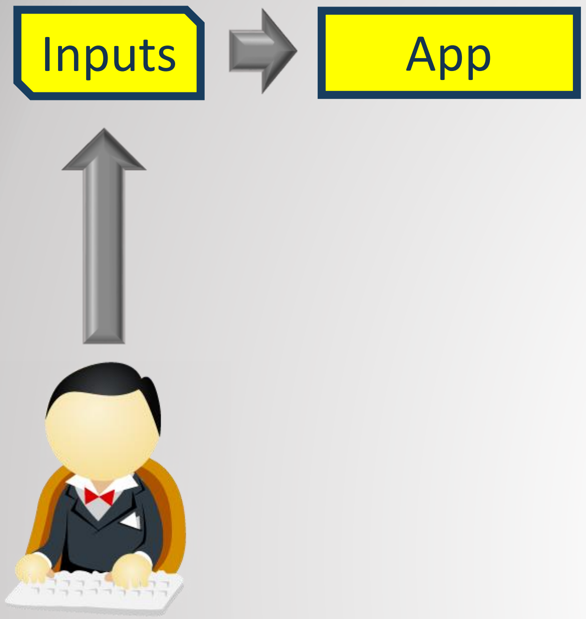
X : p×m **A**: p×k **S**: k×m

Good traces ➡ [S_{Good}]

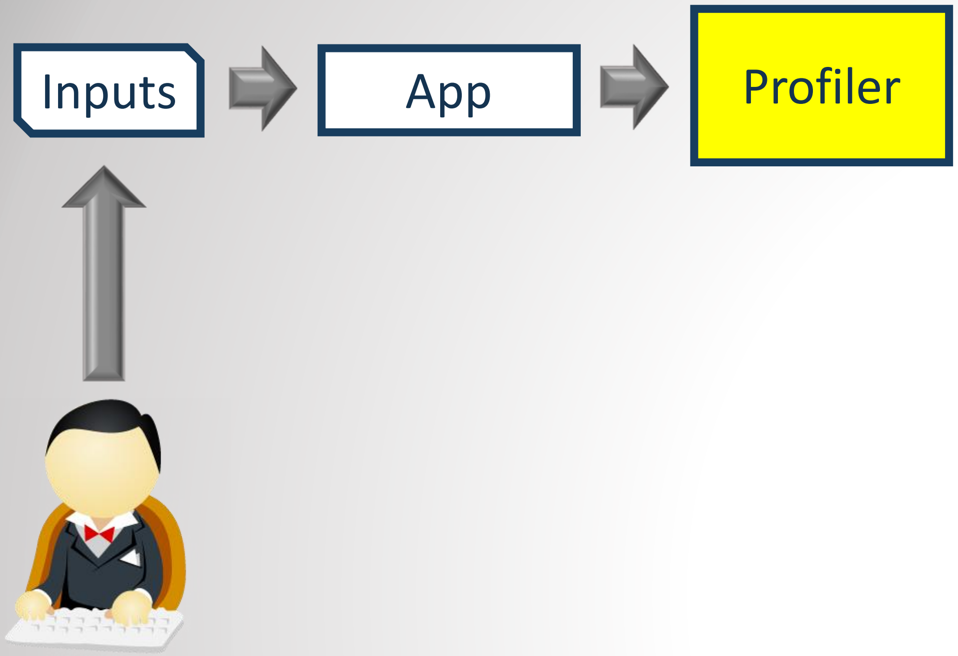
Bad traces ➡ [S_{Bad}]

For each method

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


Inputs
App


JMeter
Web-based application



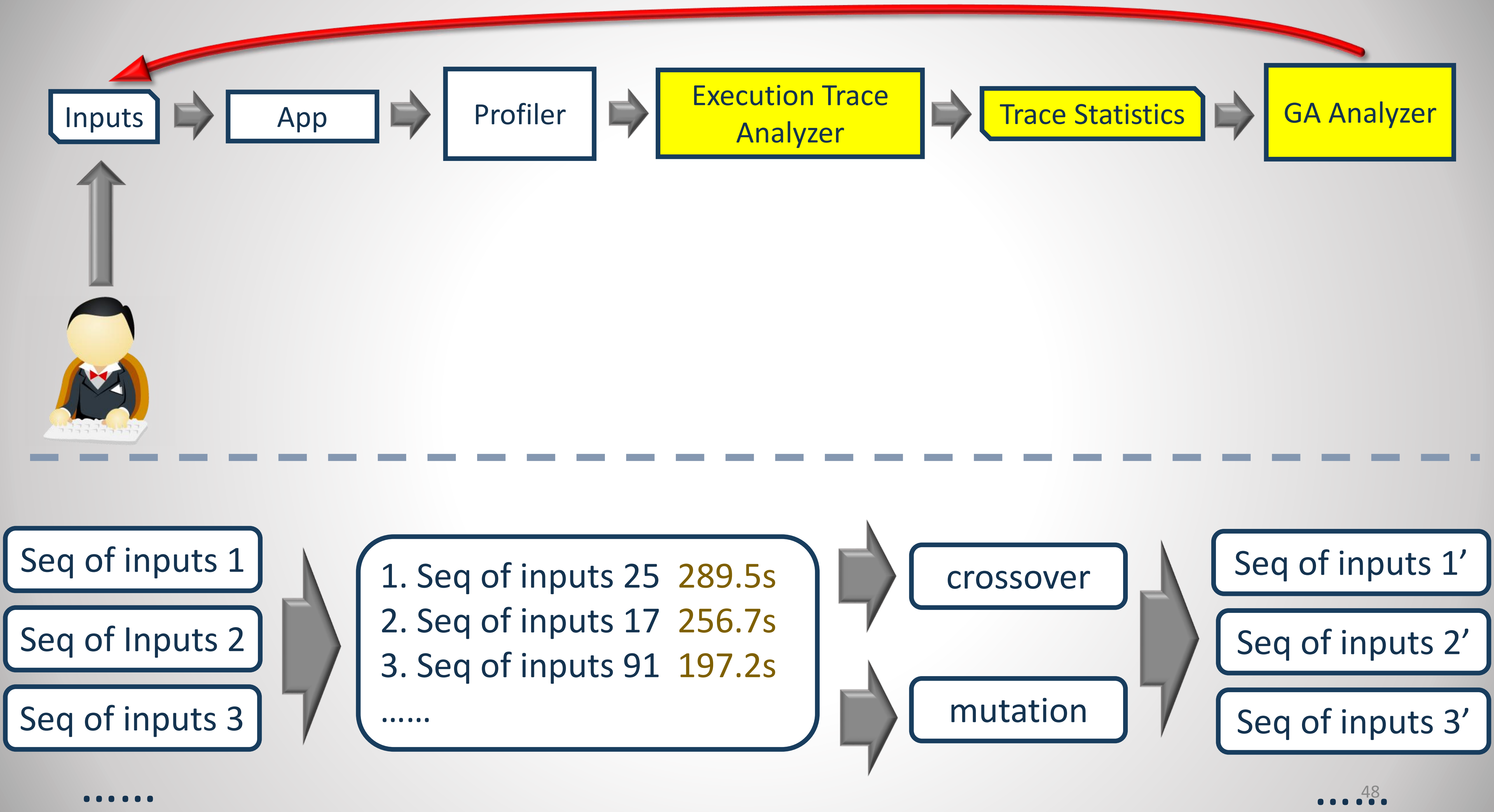
Collect execution traces

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

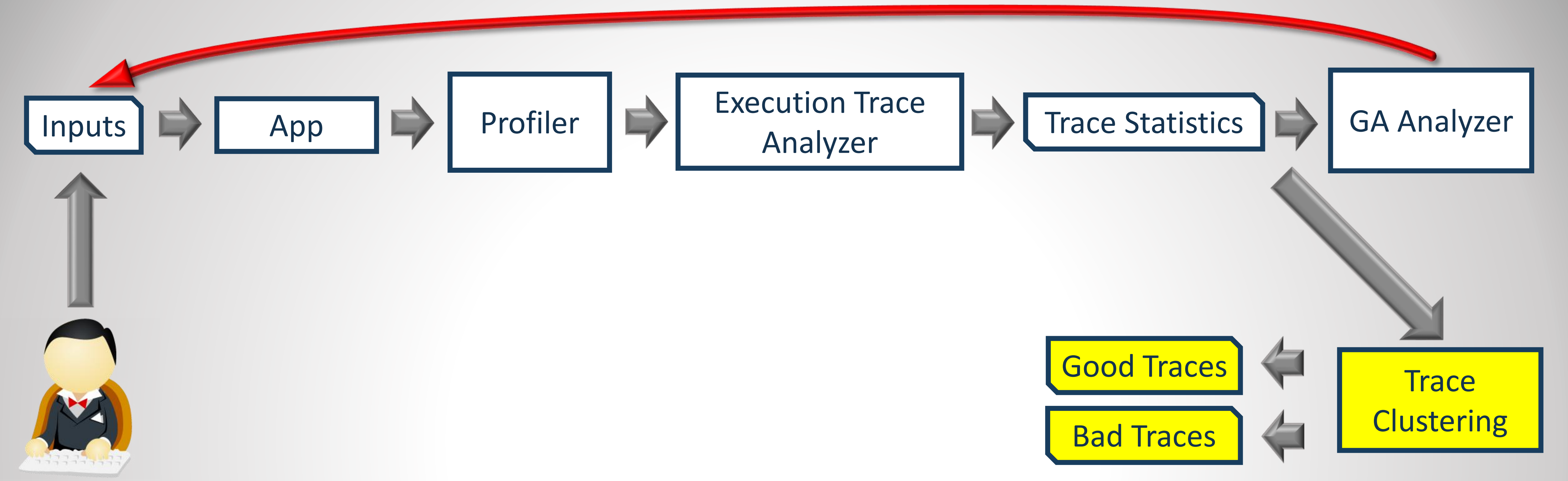
MTDRET_|_http-bio-8080-exec-7_|_131011218036560_|_fi/hut/soberit/agilefant/model/Story_|_getStoryAccesses()Ljava/util/Set;_||_

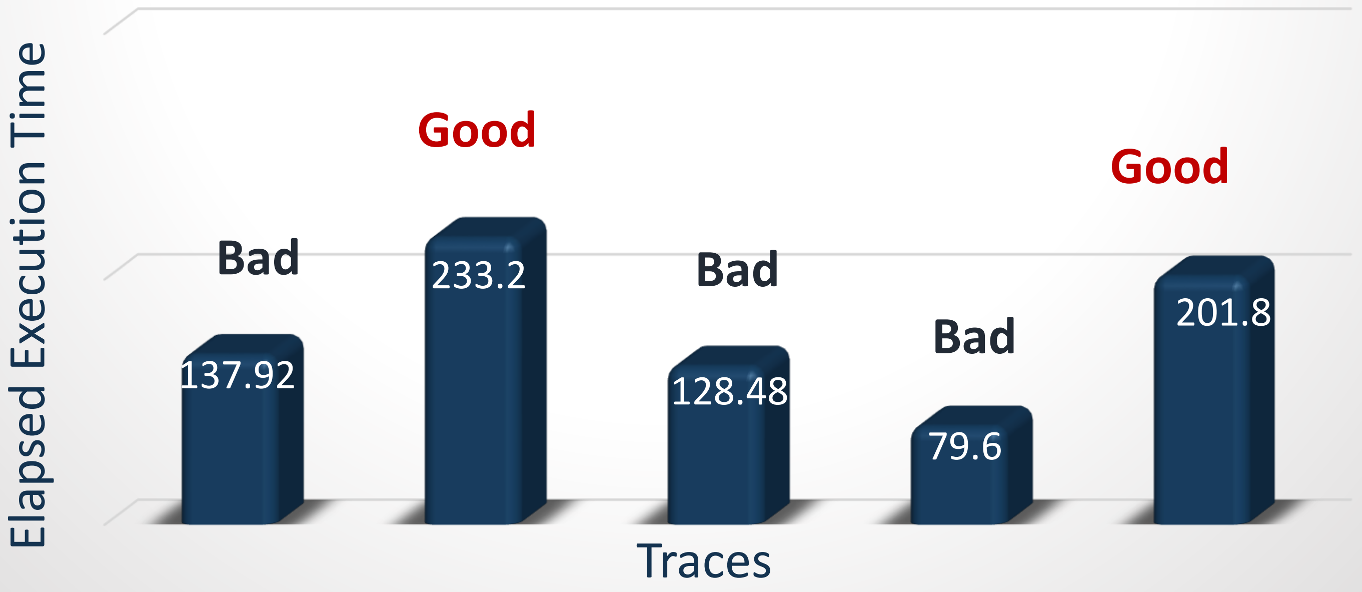
MTDENT_|_http-bio-8080-exec-7_|_131011218074491_|_fi/hut/soberit/agilefant/model/User_|_getId()I_||_

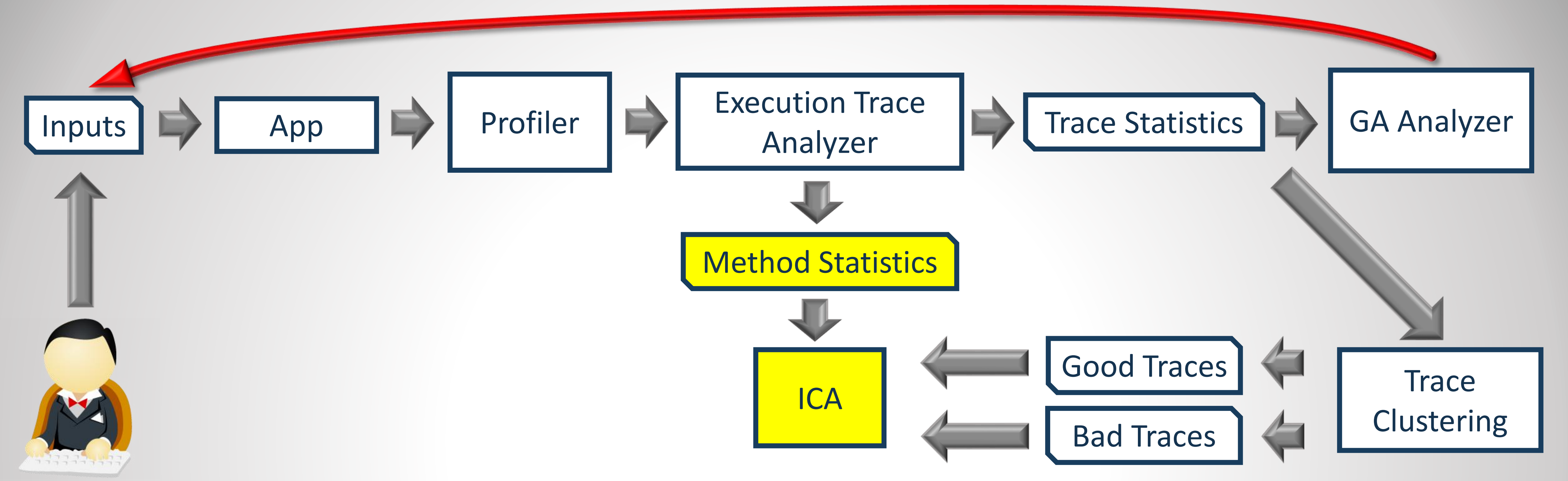
MTDRET_|_http-bio-8080-exec-7_|_131011218076276_|_fi/hut/soberit/agilefant/model/User_|_getId()I_||_



Inputs
App
Profiler
Execution Trace Analyzer
Trace Statistics
GA Analyzer
Seq of inputs 1
Seq of Inputs 2
Seq of inputs 3
……
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
Trace Clustering
Good Traces
Bad Traces

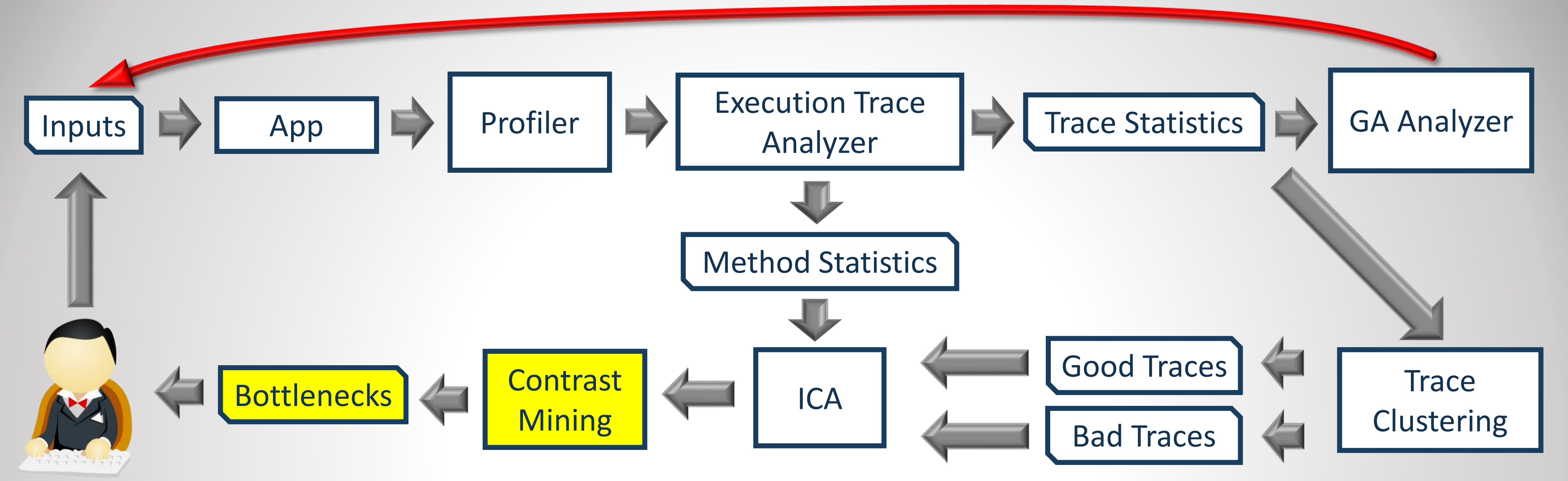

Elapsed Execution Time
Bad
137.92
Good
233.2
Bad
128.48
Bad
79.6
Good
201.8
Traces



$$\begin{array}{cc} & \text{methods} \\ \text{traces} & \begin{bmatrix} \quad & \quad \\ \quad & \quad \end{bmatrix} \end{array} = \begin{array}{cc} & \text{features} \\ \text{traces} & \begin{bmatrix} \quad \\ \quad \end{bmatrix} \end{array} \times \begin{array}{cc} & \text{methods} \\ \text{features} & \begin{bmatrix} \quad & \quad \end{bmatrix} \end{array}$$

$\mathbf{X}$: p×m $\mathbf{A}$: p×k $\mathbf{S}$: k×m

Good traces ➔ [S_{Good}] Bad traces ➔ [S_{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
4. Agilefant.model.Backlog.setChildren()	7.932
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

AGILEFANT.ORG - THE SIMPLEST BACKLOG TOOL THAT SCALES

Dell DVD Store

DVD Store

Selected Items: specify quantity desired and click Update; click Purchase when finished

Item	Quantity	Title	Actor	Price	Remove From Order?
1	1	ACE CRANES	VAL PENN	12,99	☐
2	2	AIRPORT SEABISCUIT	JUDE CRAWFORD	15,99	☐
			Subtotal	44,97	
			Tax (8,25%)	3,71	
			Total	48,68	

Update and Recalculate Total

Purchase

Thank You for Visiting the DVD Store!

Copyright © 2005 Dell

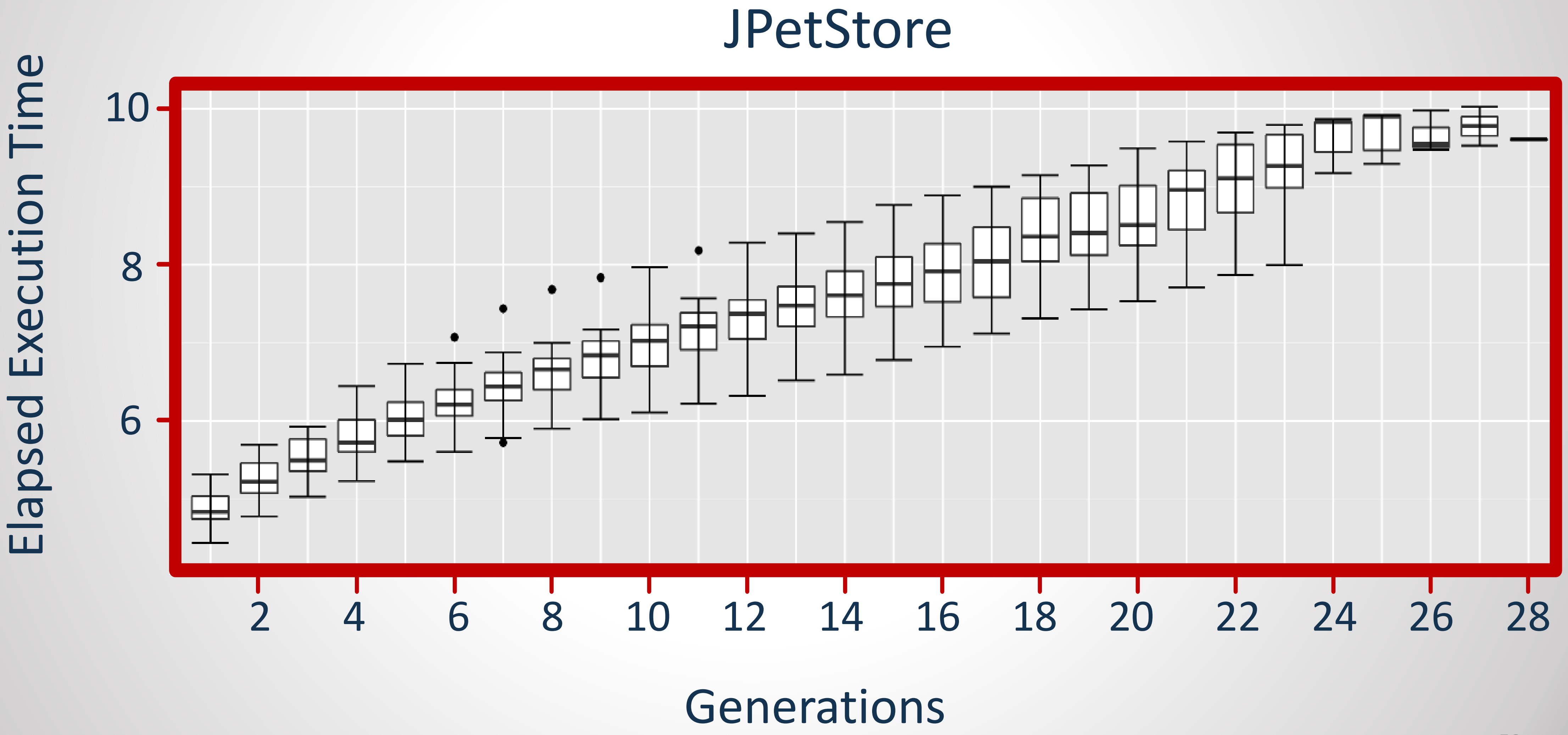
Experimental Design

5 users

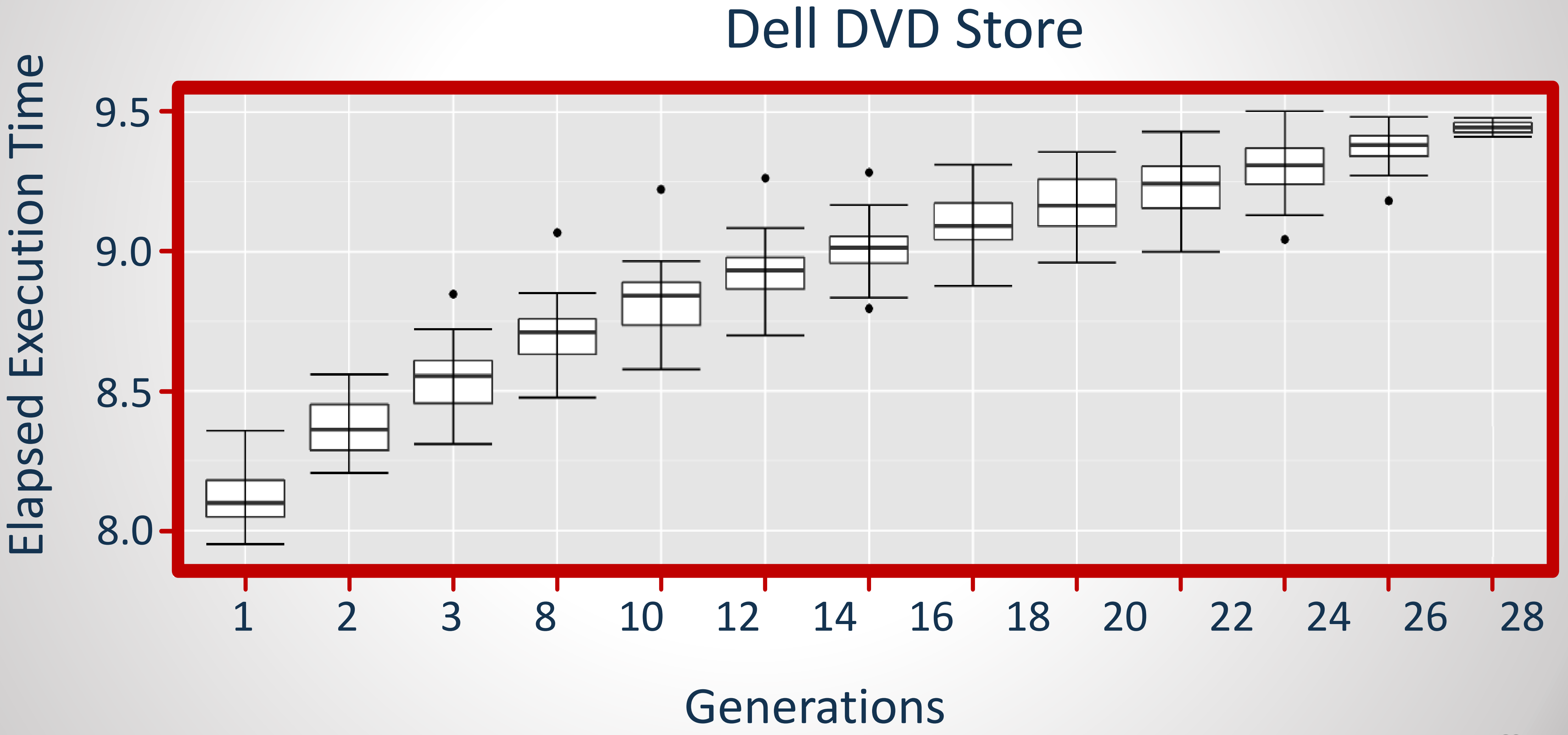
50 URLs per user

30 times to repeat

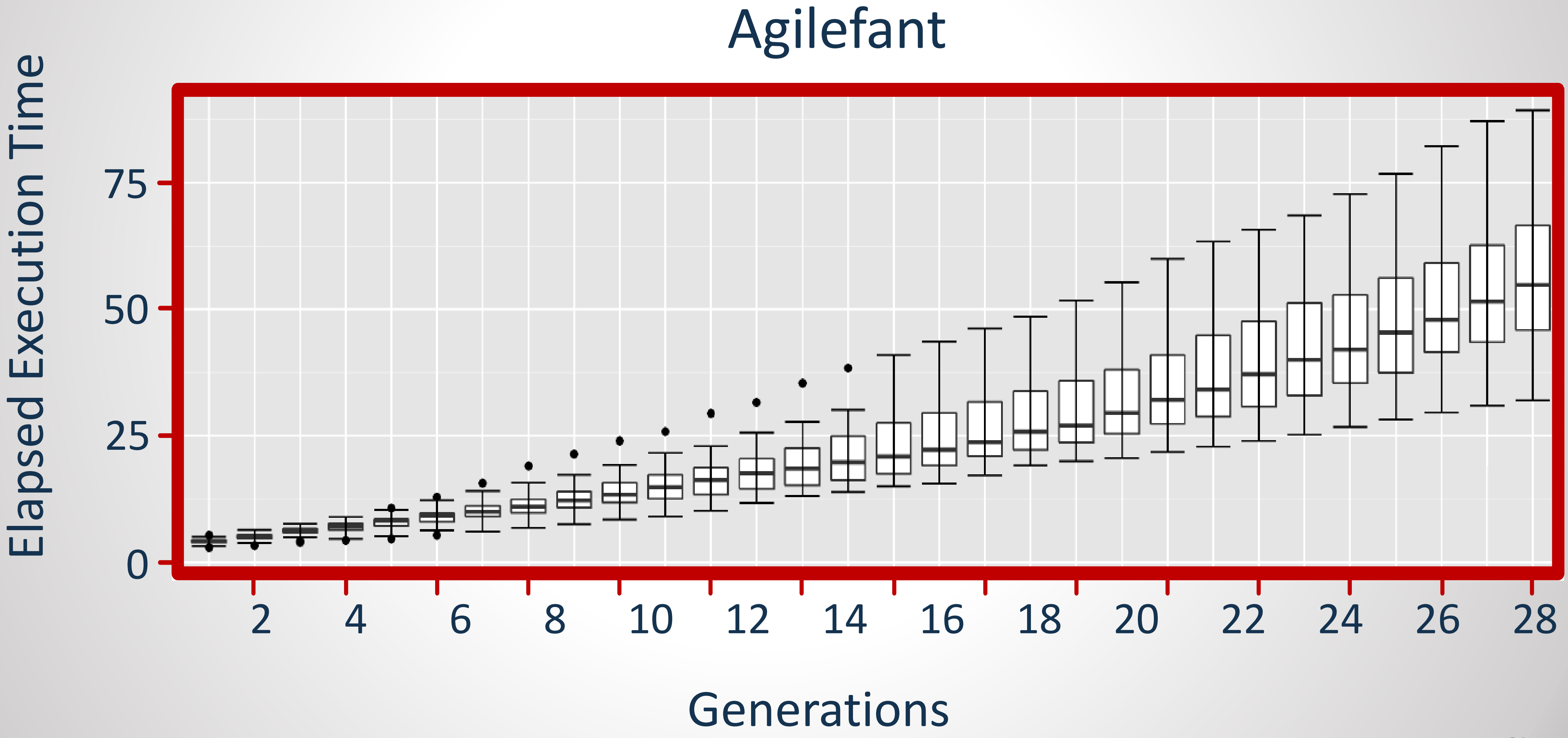
RQ1 – Finding Bottleneck-Specific Inputs



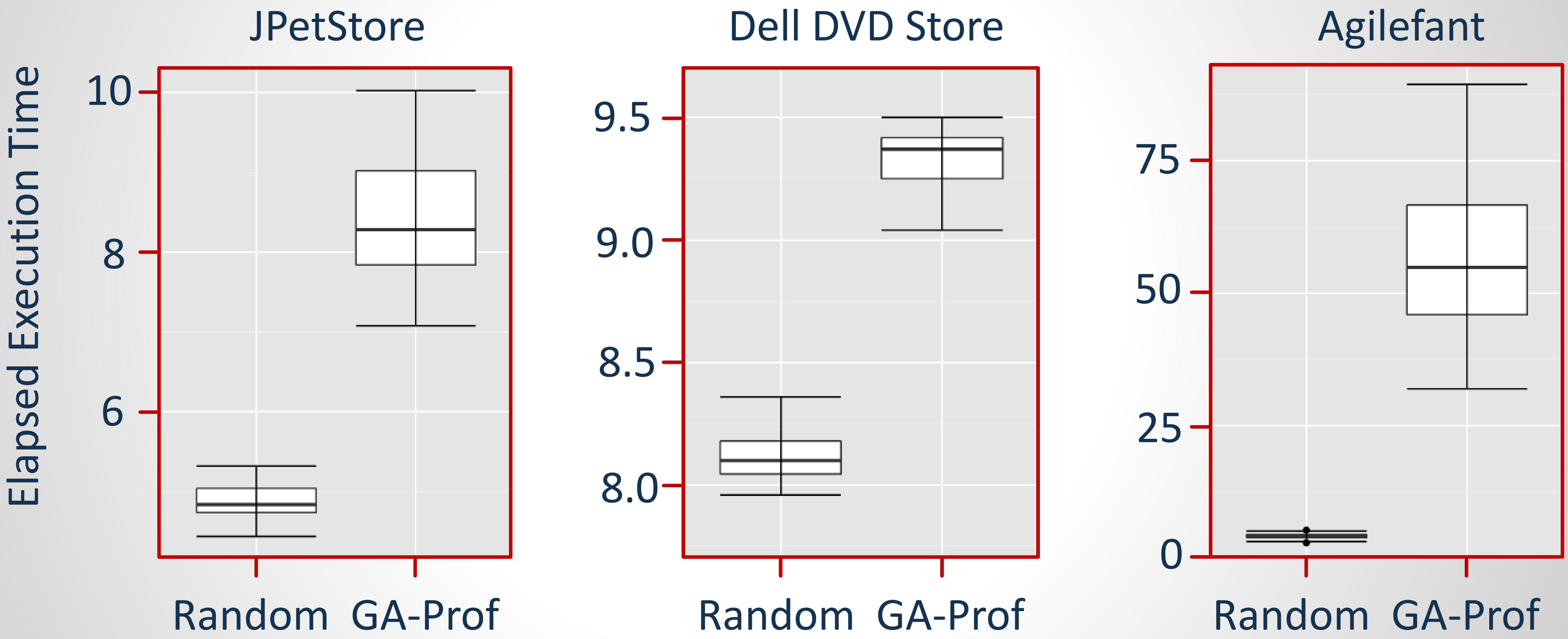
RQ1 – Finding Bottleneck-Specific Inputs



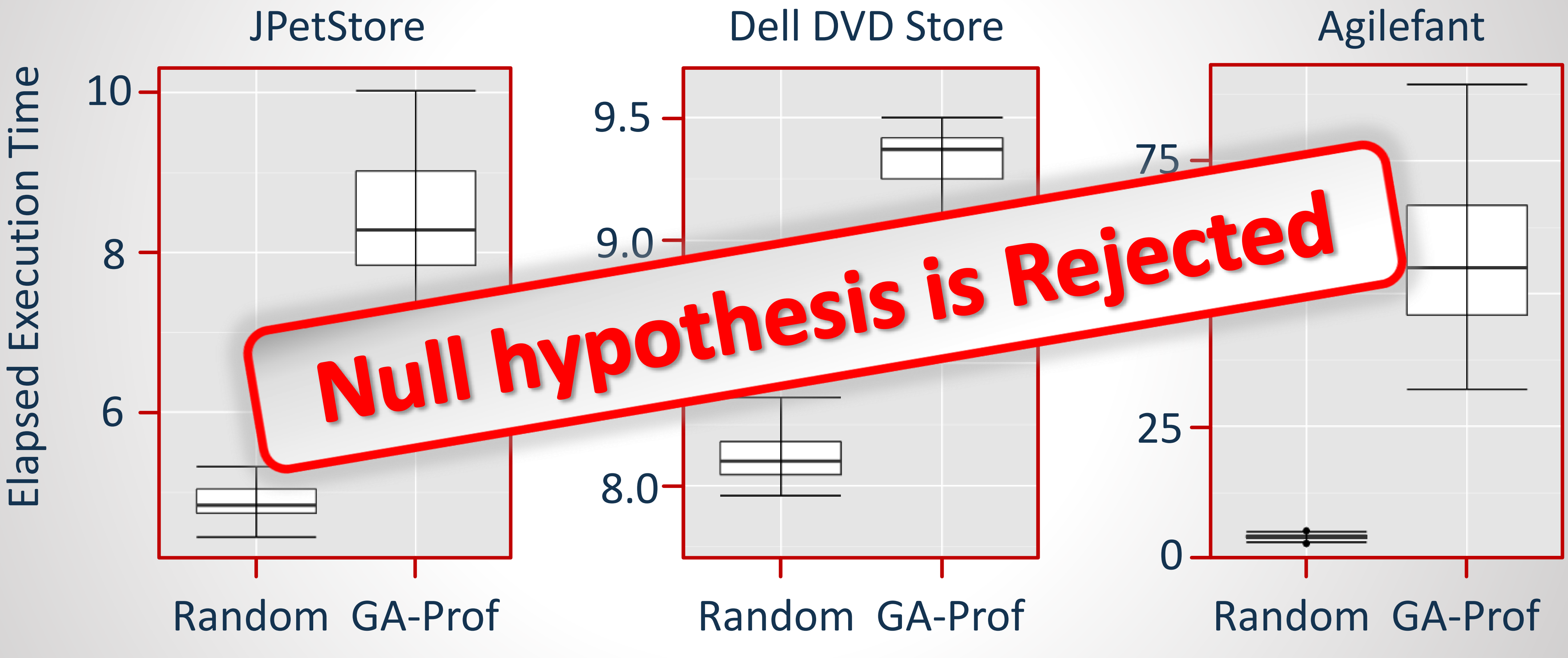
RQ1 – Finding Bottleneck-Specific Inputs



RQ1 – Finding Bottleneck-Specific Inputs

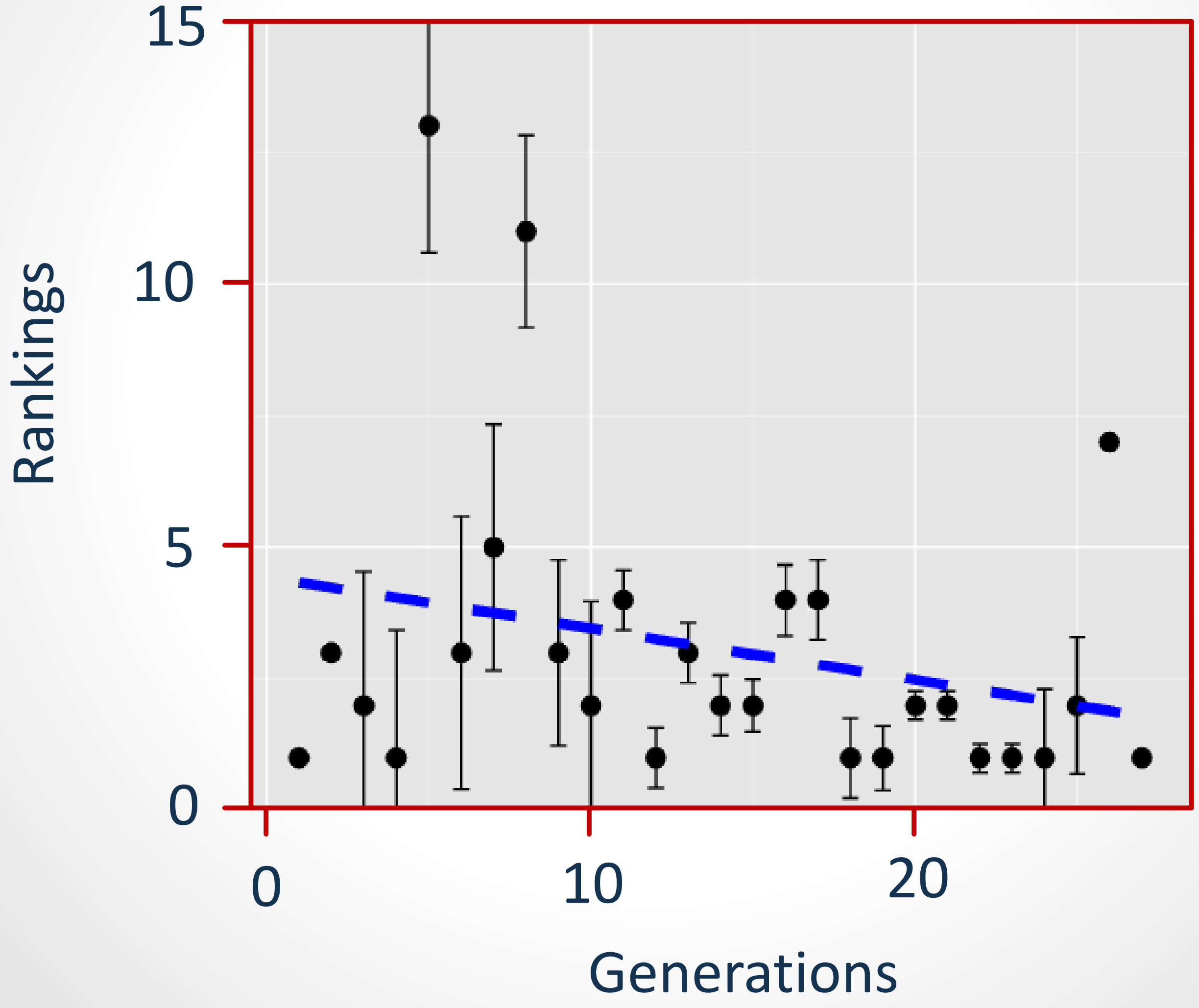


RQ1 – Finding Bottleneck-Specific Inputs



$p = 1.5e - 21$ $p = 2.9e - 30$ $p = 6.4e - 17$

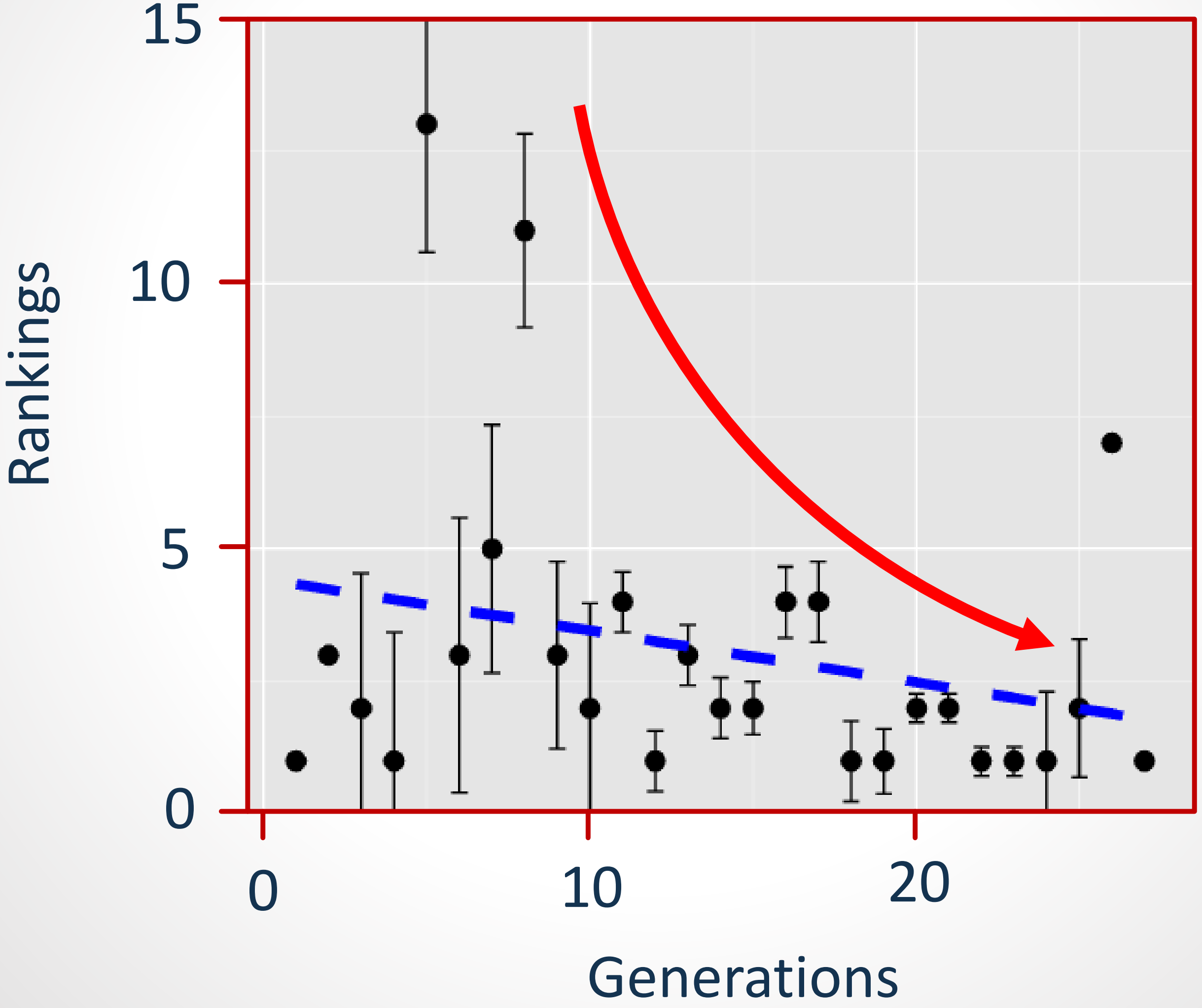
RQ2 – Finding Injected Bottlenecks



Injected bottleneck:

Jpetstore.domain.Product.getName()

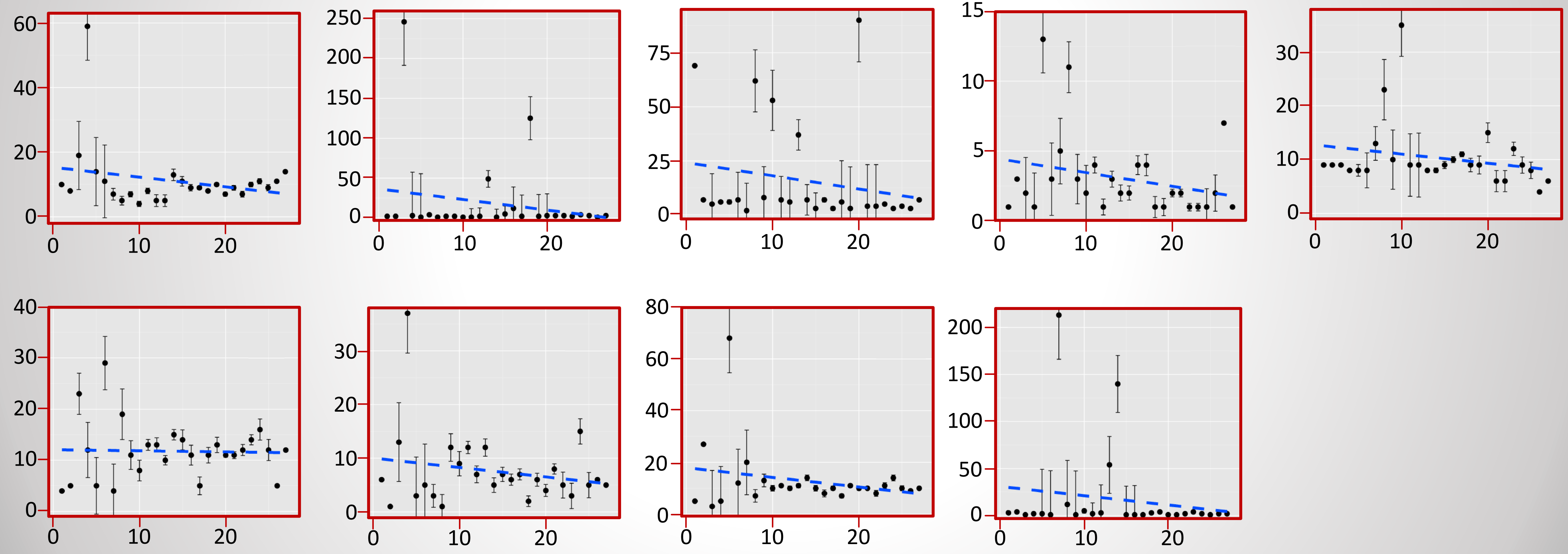
RQ2 – Finding Injected Bottlenecks



Injected bottleneck:

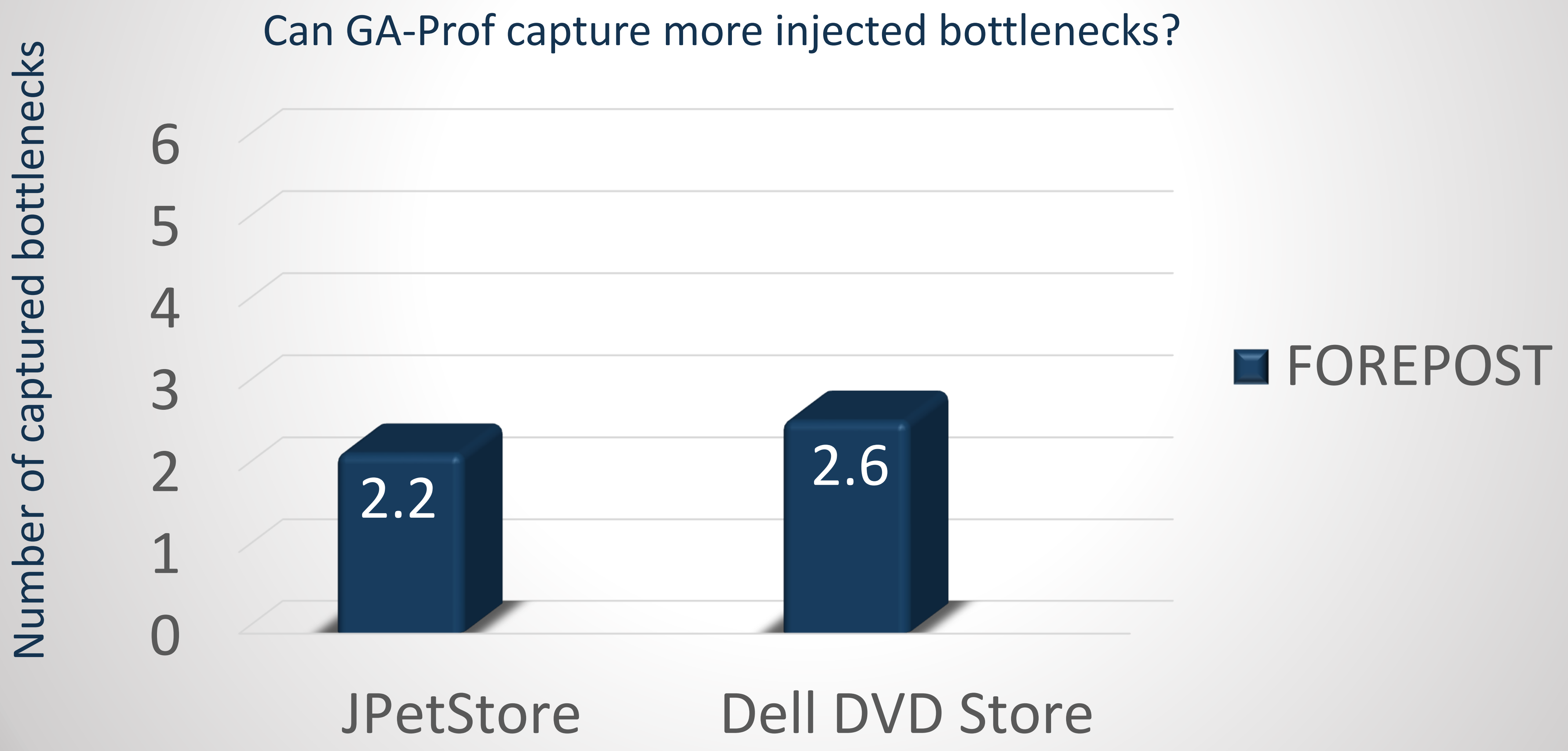
Jpetstore.domain.Product.getName()

RQ2 – Finding Injected Bottlenecks

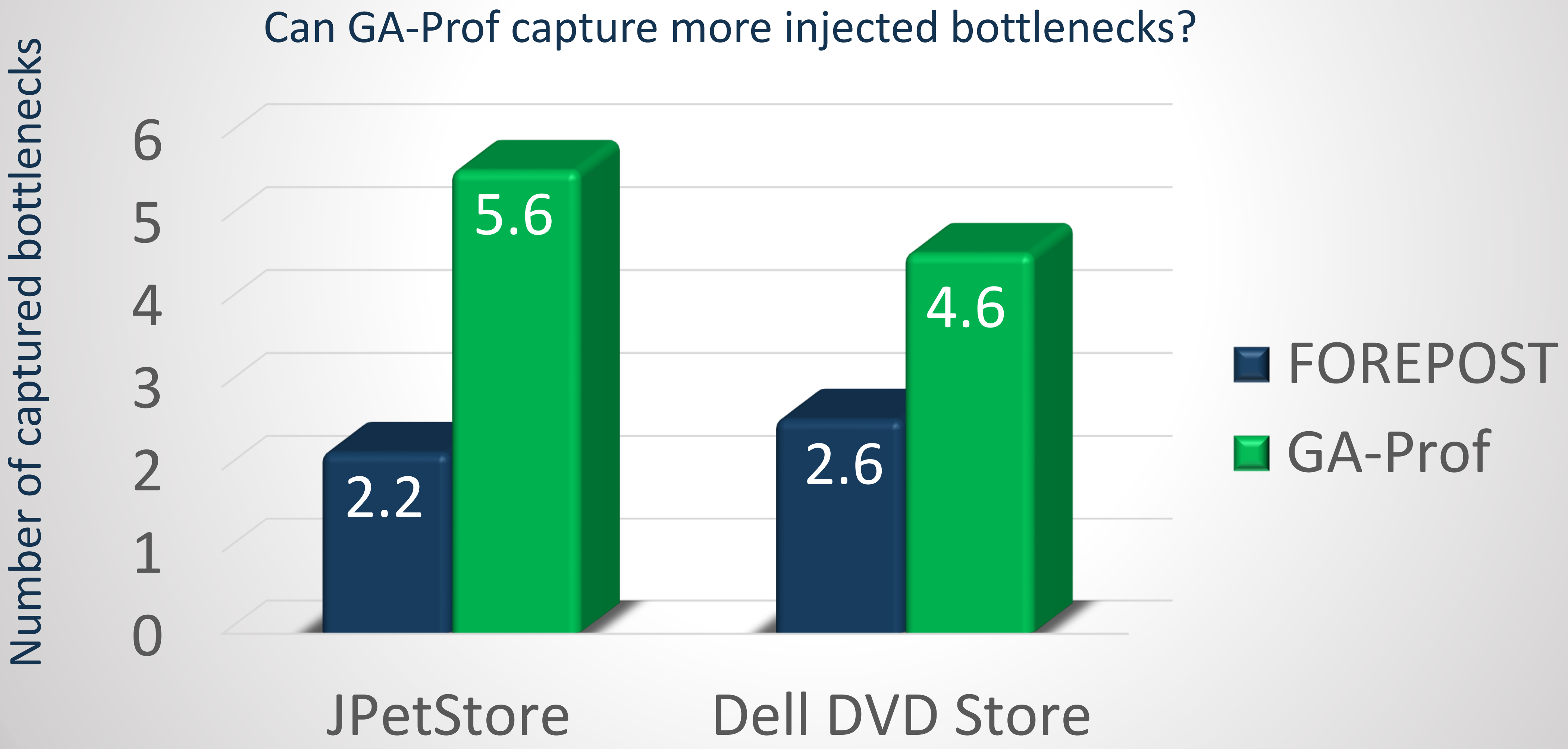


Rankings of Nine Injected bottlenecks X axis – rankings, Y axis – number of generations

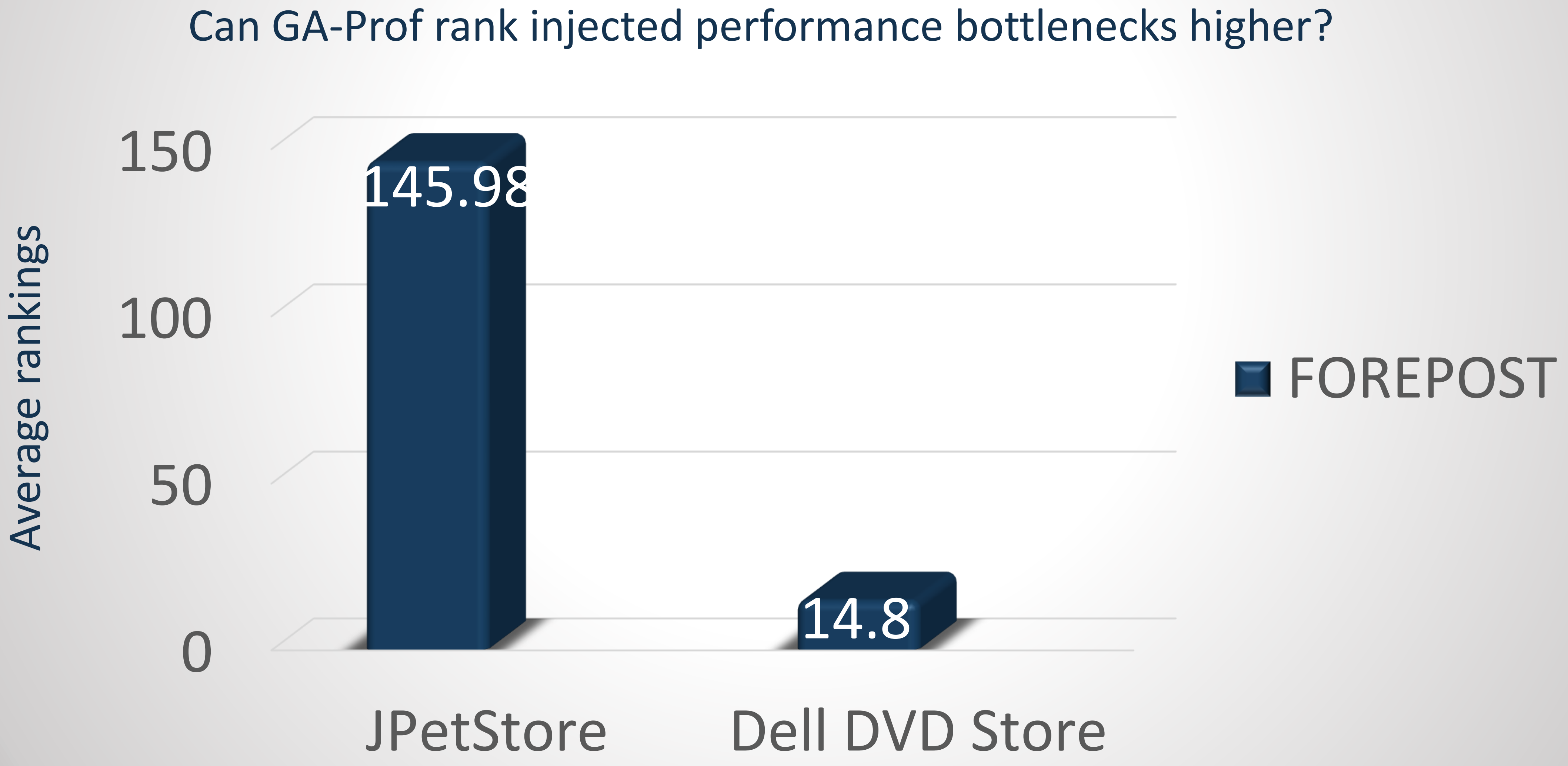
RQ3 – GA-Prof vs. FOREPOST



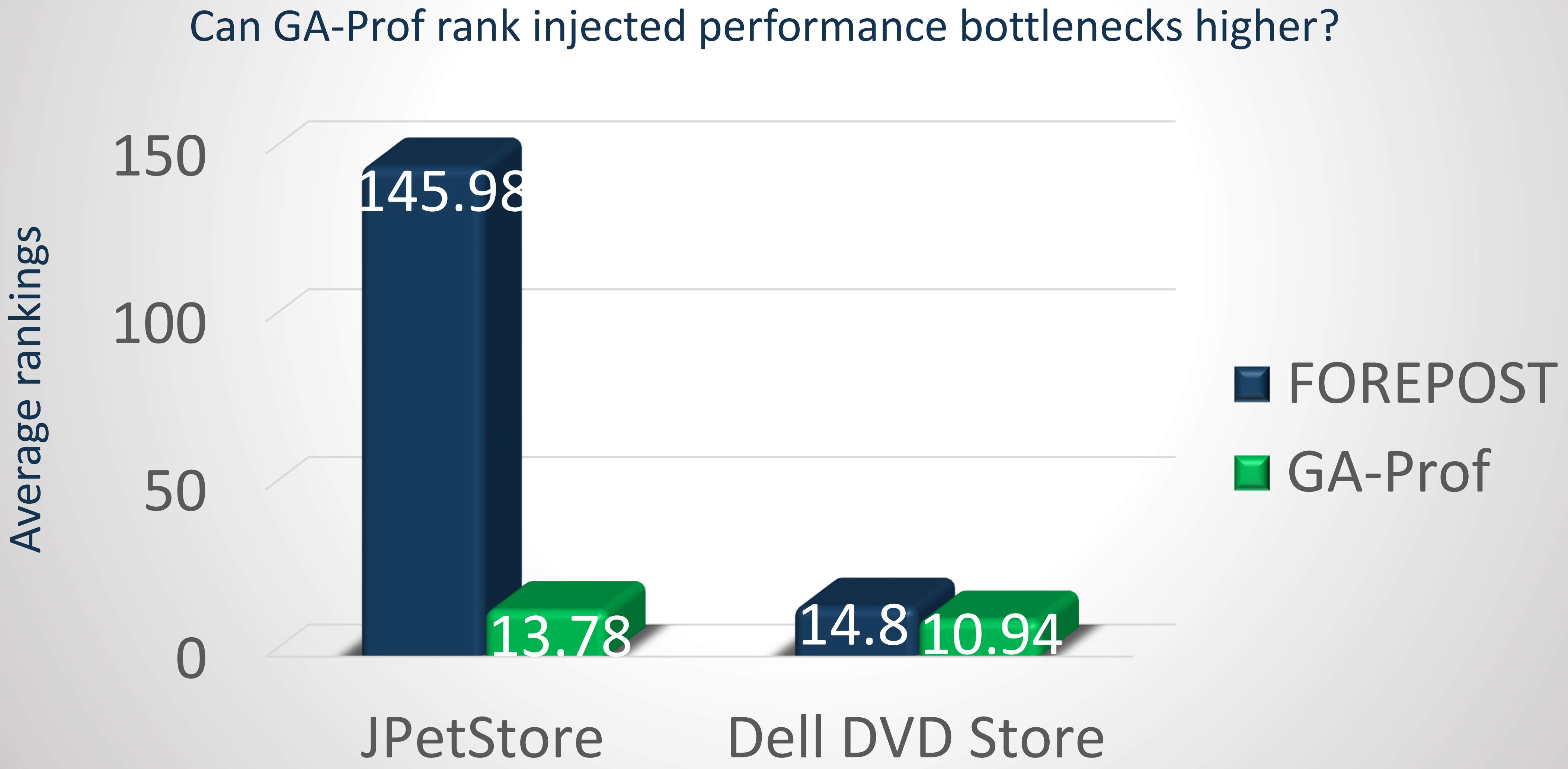
RQ3 – GA-Prof vs. FOREPOST



RQ3 – GA-Prof vs. FOREPOST



RQ3 – GA-Prof vs. FOREPOST



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,denys@cs.wm.edu

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

ABSTRACT

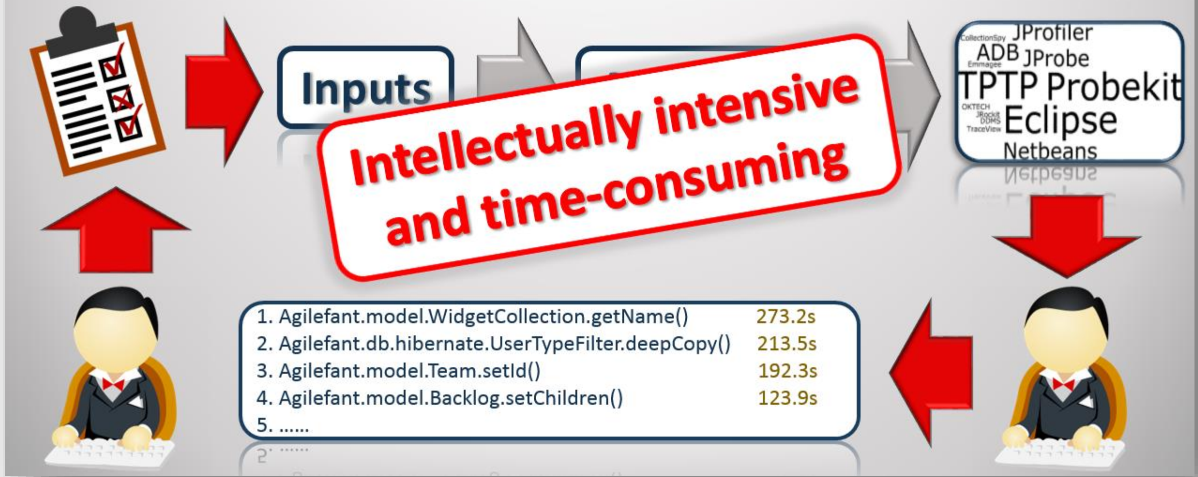
Application profiling is an important performance analysis technique, when an application under test is analyzed dynamically to determine its space and time complexities and the usage of its instructions. A big and important challenge is to profile nontrivial web applications with large numbers of combinations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

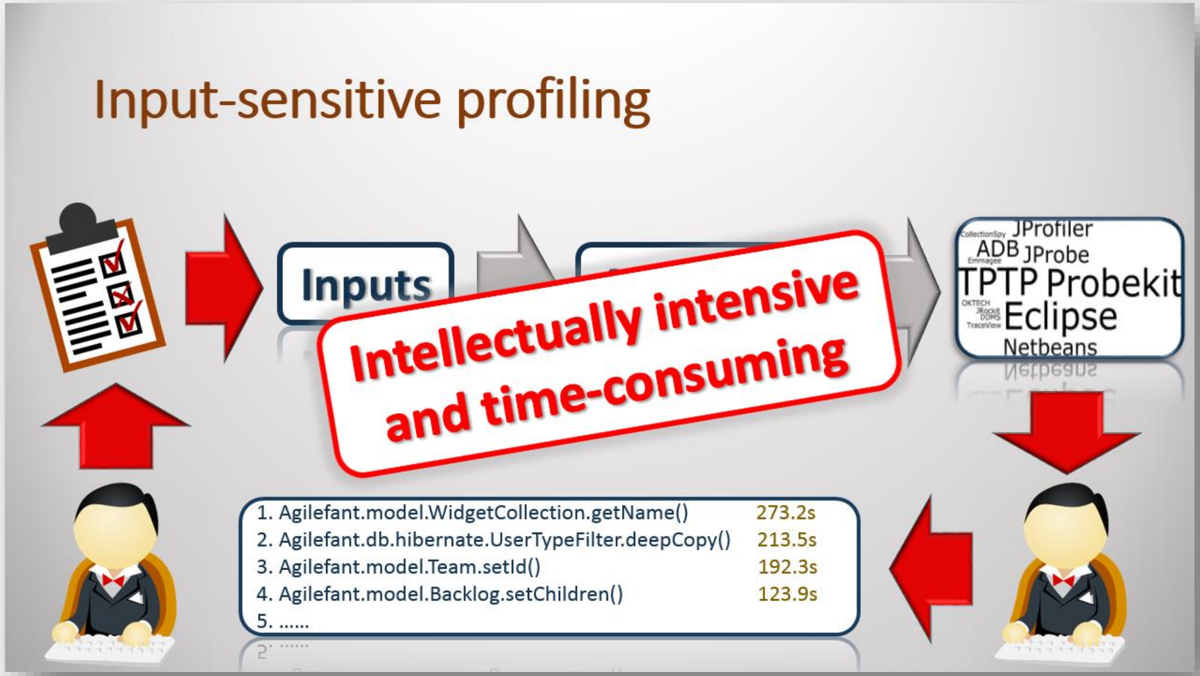
1. INTRODUCTION

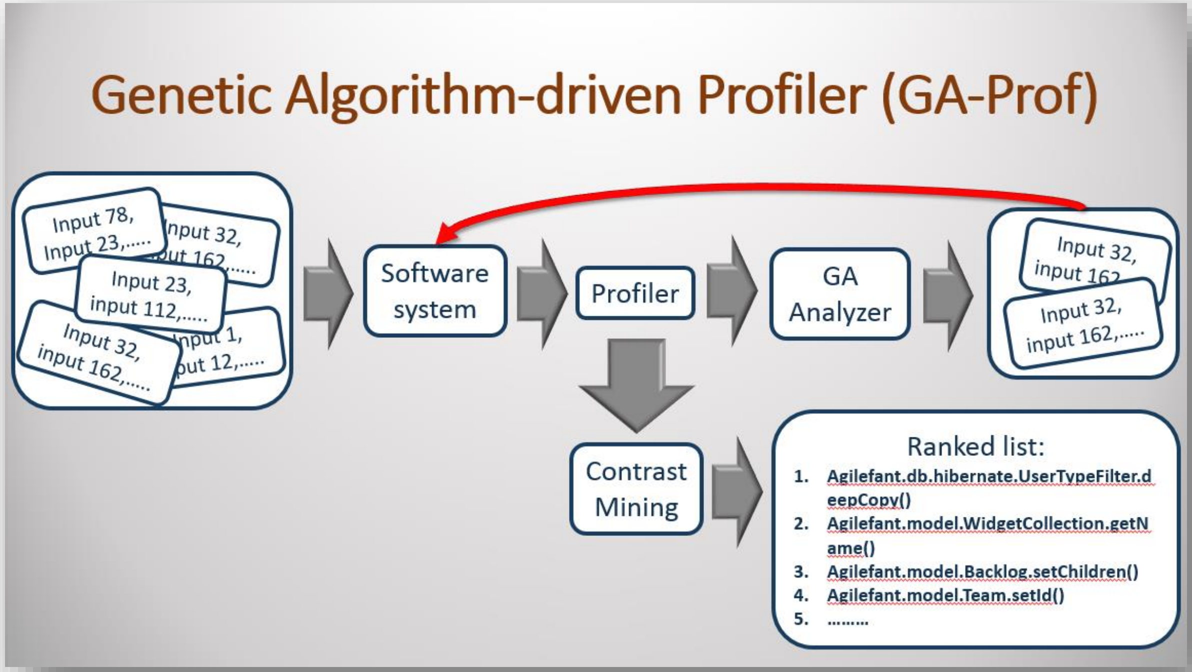
Improving performance of software applications is one of the most important tasks in software evolution and maintenance [16]. Software engineers make performance enhancements routinely during perfective maintenance [55] when they use exploratory random performance testing [25, 11] to identify methods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

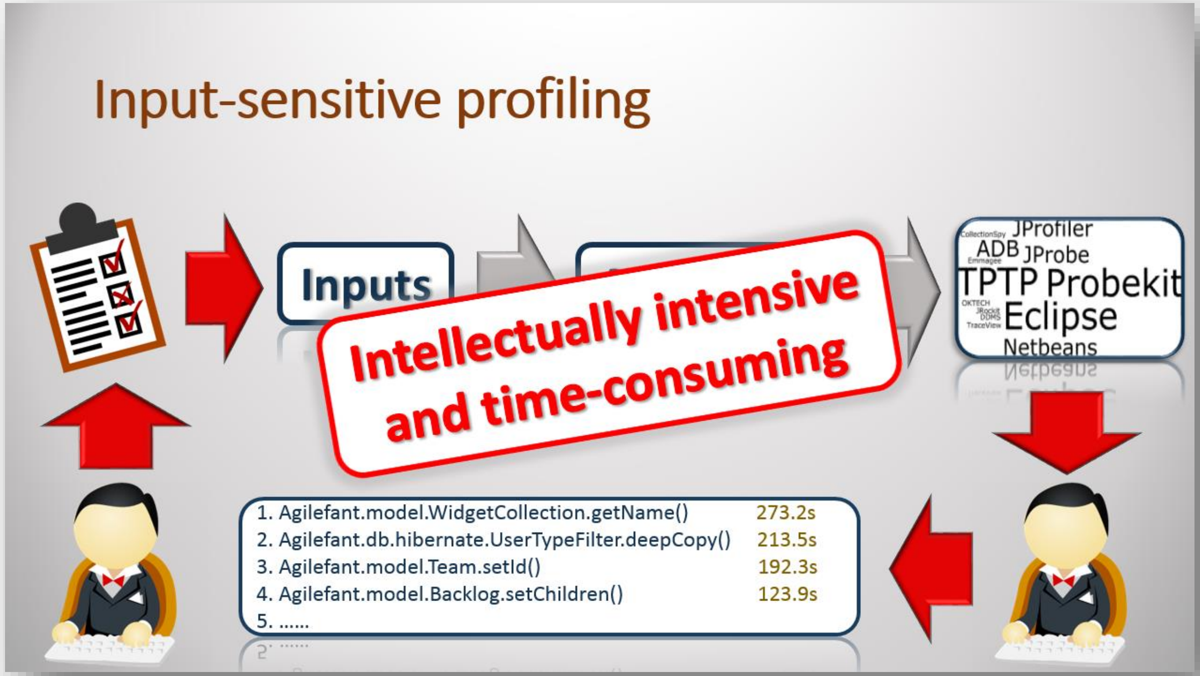
Online appendix: http://www.cs.wm.edu/semeru/data/ISSTA15-GAProf/

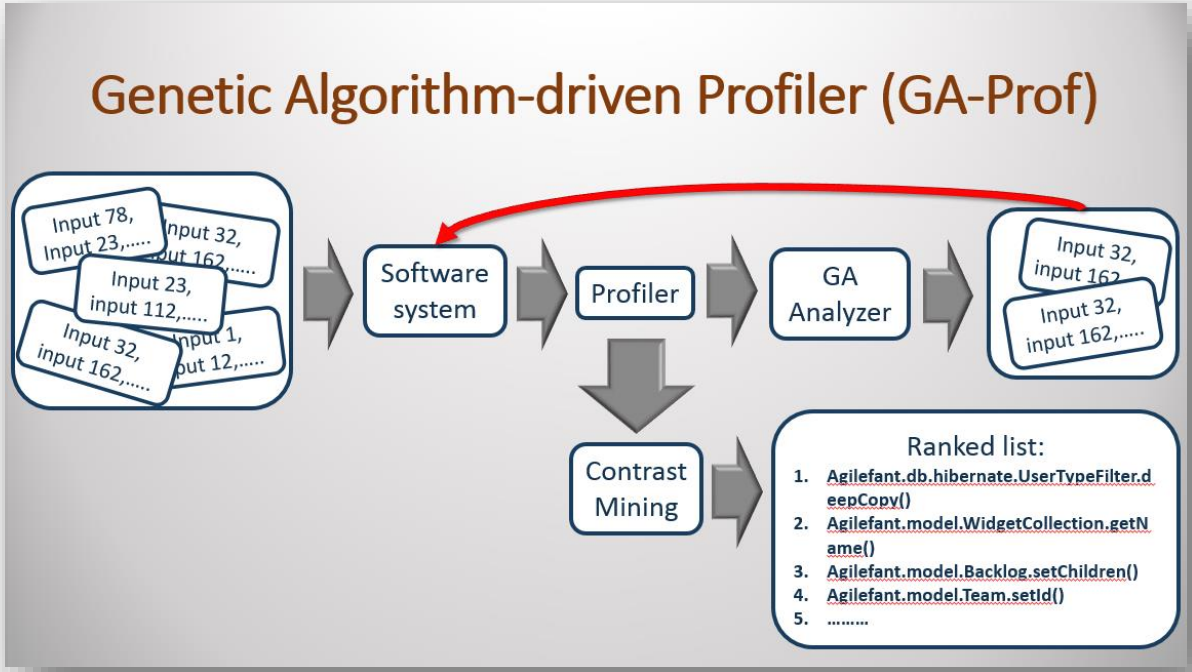
Input-sensitive profiling




Input-sensitive profiling
Inputs
Intellectually intensive and time-consuming
JProfiler
ADB
JProbe
TPTP Probekit
Eclipse
Netbeans
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)
Input 78, Input 23,.....
Input 23, input 112,.....
Input 32, input 162,.....
Software system
Profiler
GA Analyzer
Input 32, input 162,.....
Contrast Mining
Ranked list:
1. Agilefant.db.hibernate.UserTypeFilter.deepCopy()
2. Agilefant.model.WidgetCollection.getName()
3. Agilefant.model.Backlog.setChildren()
4. Agilefant.model.Team.setId()
5. ………

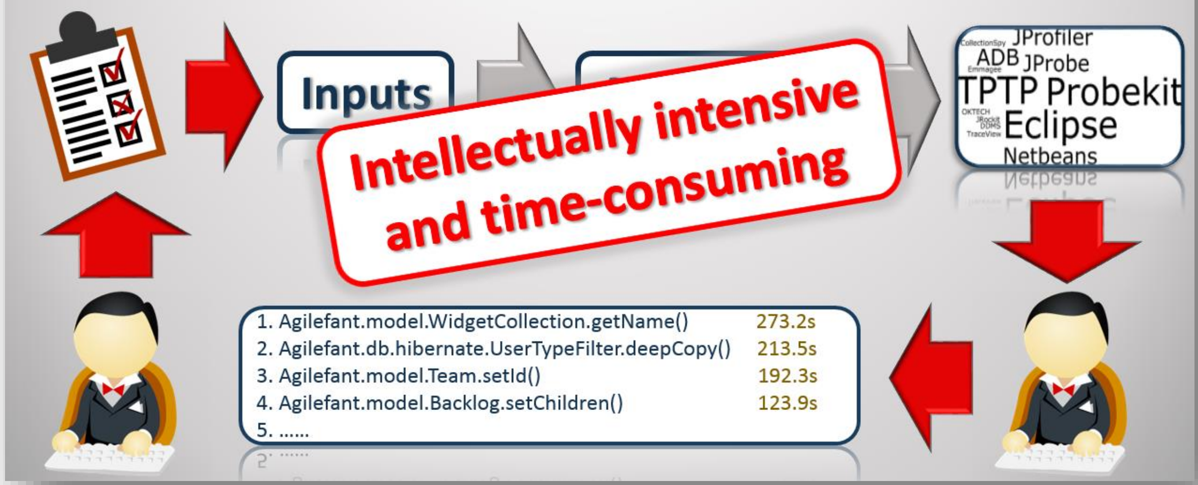




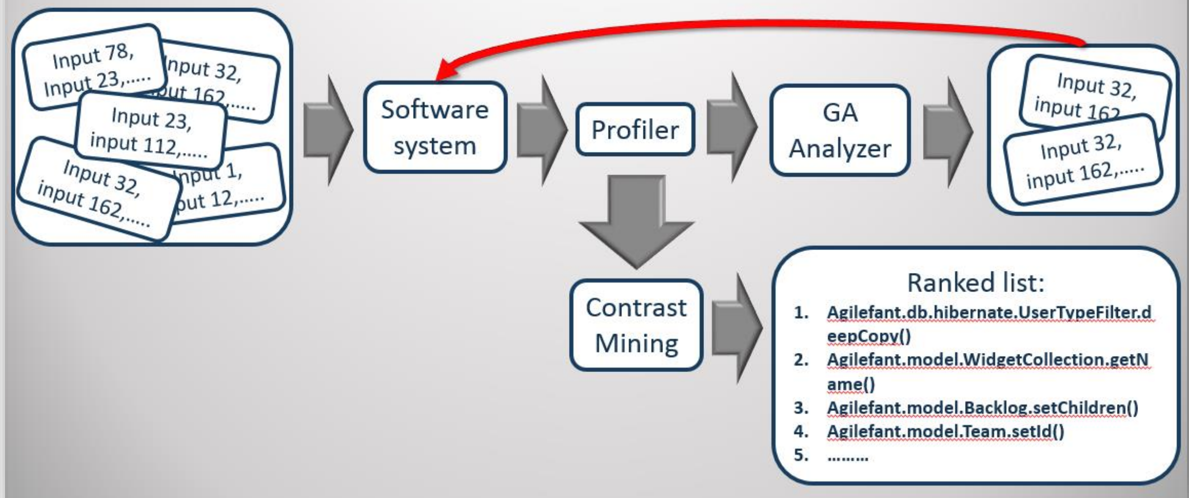
Research Questions (RQs)

- 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

Input-sensitive profiling



Genetic Algorithm-driven Profiler (GA-Prof)



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