CloPlag

A Study of Effects of Code Obfuscation to Code Similarity Detection Tools

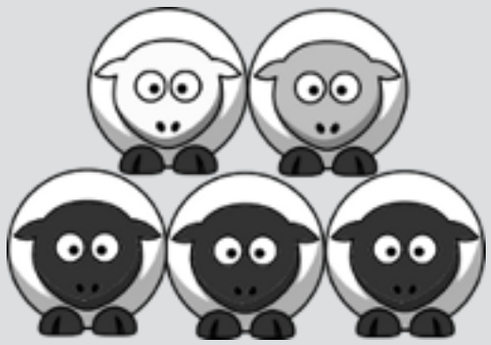
Chaiyong Ragkhitwetsagul, Jens Krinke, Albert Cabré Juan

CREST

UCL

Cloned Code vs Plagiarised Code

- A result from source code reuse by copying and pasting [maybe with some modifications]
- Segments of code which are identical or similar
- Code maintenance and management
- In some cases, code cloning may violate software license[1]



- Created in a similar way as code clones but with different intention
- Source code plagiarism violates academic regulations
- Oracle vs Google law suit[2]

[1] A. Monden, S. Okahara, Y. Manabe, and K. Matsumoto, "Guilty or Not Guilty: Using Clone Metrics to Determine Open Source Licensing Violations," IEEE Software, vol. 28, no. 2, pp. 42–47, 2011.
[2] http://www.mondaq.com/unitedstates/x/271942/

What is Obfuscation?

- Modifying a program while preserving its semantics
- Can be achieved at 2 levels:
  - Source code
  - Byte code

```
@P=split//,".URRUU\c8R";@d=split//,"\nrekcah xinU / lreP rehtona tsuJ";sub p{
@p{"r$p","u$p"}=(P,P);pipe"r$p","u$p";++$p;($q*=2)+=$f=!fork;map{$P=$P[$f^ord
($p{$_})&6];$p{$_}=/ ^$P/ix?$P:close$_}keys%p}p;p;p;p;p;map{$p{$_}=~/^[P.]/&&
close$_}%p;wait until$?;map{/^r/&&<$_>}%p;$_=$d[$q];sleep rand(2)if/\S/;print
```

Research Questions

RQ1: how do current detection tools perform against code obfuscation?

RQ2: what is the best parameter settings and similarity threshold of each tool?

RQ3: how do compilation and decompilation facilitate the detection process?

RQ4: can we apply the best parameters and threshold to other datasets effectively?

Overview of the Empirical Study

- Java programs are obfuscated at:
  - Source code level
  - Byte code level
  - Combination of both
- Several similarity detection tools are applied to the data set
- Varying the settings and threshold of each tool
- Measure performance of each tool

Tools

Obfuscators

ARTIFICE
ProGuard

Decompilers

Procyon
Krakatau

Detectors

Clone
SW plagiarism
Compression
Others

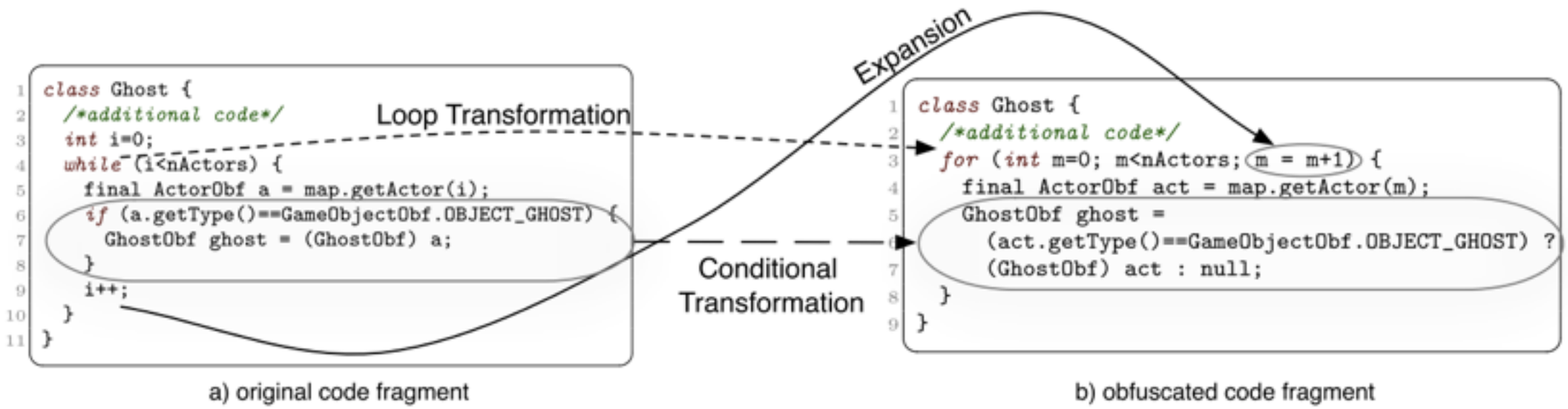
Obfuscators

ARTIFICE

- Source code level
- Renaming, changing loops & conditional statements, changing increment/ decrement statements

ProGuard

- Bytecode level
- Rename classes, fields, variables to short, meaningless



Schulze, S., & Meyer, D. (2013). On the robustness of clone detection to code obfuscation. 2013 7th International Workshop on Software Clones (IWSC)

Detectors

Clone detectors

CCFinderX
iClones
Simian, NiCad
Deckard

Plagiarism detectors

JPlag
Sherlock, Plaggie
Sim

ncd-bzlib
7zncd-BZip2
Inclusion

Compression

diff, bsdiff
py-difflib
py-sklearn.cosine_similarity

Others

* Totally 21 tools
** All tools have to report similarity values (0 - 100)

Test Case 1

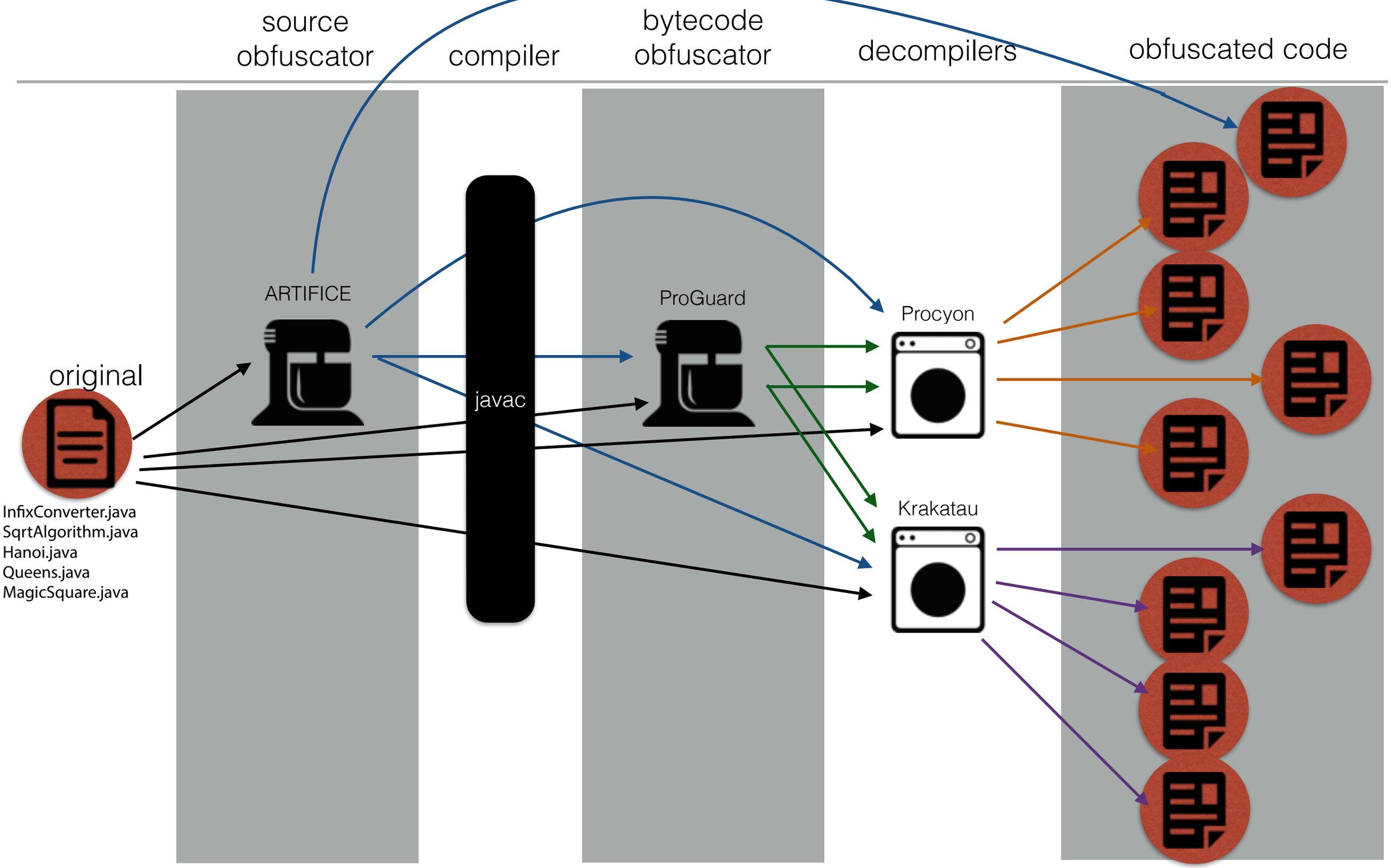
- RQ1: how do current detection tools perform against code obfuscation?
- RQ2: what is the best parameter settings and similarity threshold of each tool?
- A series of small Java programs
  - InfixConverter
  - SqrtAlgorithm
  - Hanoi
  - Queens
  - MagicSquare

```
public static String InfixToPostfixConvert ( String infixBuffer ) {
    int priority = 0;
    String postfixBuffer = "";
    Stack s1 = new Stack();
    for ( int i = 0; i < infixBuffer.length(); i++ ) {
        char ch = infixBuffer.charAt ( i );
        if ( ch == '+' || ch == '-' || ch == '*' || ch == '/' ) {
            if ( s1.size() <= 0 ) {
                s1.push ( ch );
            } else {
                Character chTop = ( Character ) s1.peek();
                if ( chTop == '*' || chTop == '/' ) {
                    priority = 1;
                } else {
                    priority = 0;
                }
                if ( priority == 1 ) {
                    if ( ch == '+' || ch == '-' ) {
                        postfixBuffer += s1.pop();
                        i--;
                    } else {
                        postfixBuffer += s1.pop();
                        i--;
                    }
                } else {
                    if ( ch == '+' || ch == '-' ) {
                        postfixBuffer += s1.pop();
                        s1.push ( ch );
                    } else {
                        s1.push ( ch );
                    }
                }
            }
        } else {
            postfixBuffer += ch;
        }
    }
    int len = s1.size();
    for ( int j = 0; j < len; j++ ) {
        postfixBuffer += s1.pop();
    }
    return postfixBuffer;
}
```

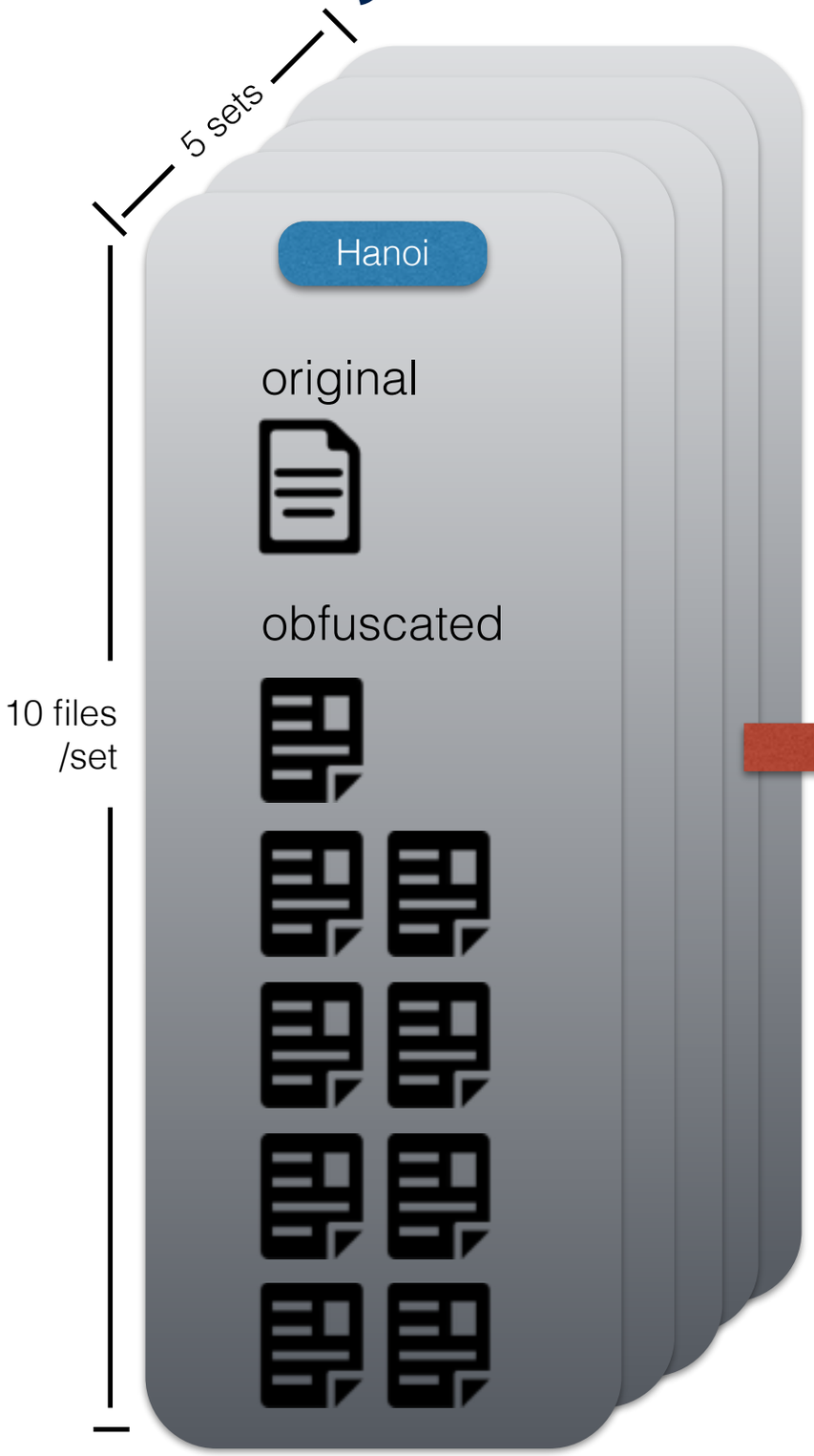
Test Data Preparation

obfuscated source code

to be used in detection phase



Similarity Calculation



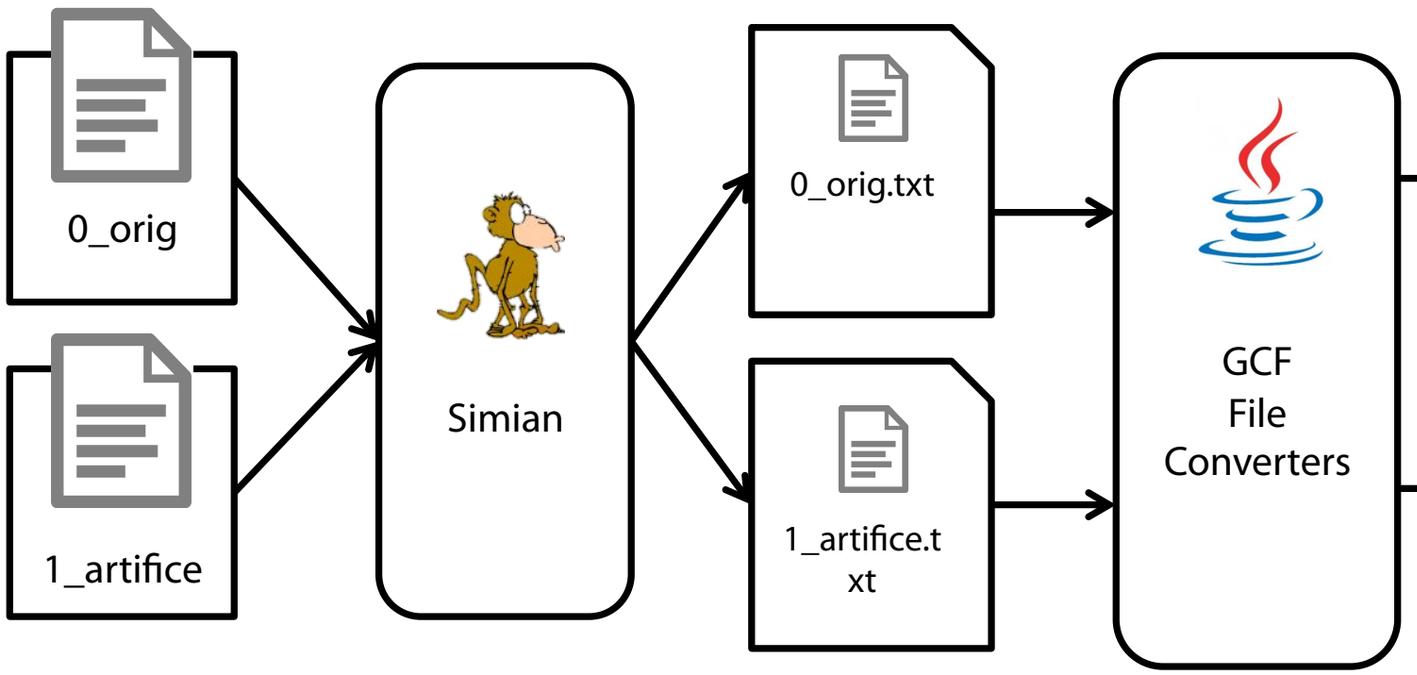
Detection tools

ccfx*
jplag*
sim
…
py-difflib

similarity report

* Most tools have different parameter settings which can strongly affect the results

Similarity Calculation for Unsupported Tools




```
- <CloneClasses>
  - <CloneClass>
      <ID>1</ID>
    - <Clone>
      - <Fragment>
          <File>hanoi/0_orig/Main.java</File>
          <Start>69</Start>
          <End>84</End>
        </Fragment>
      </Clone>
    - <Clone>
      - <Fragment>
          <File>hanoi/1_artifice/Main.java</File>
          <Start>72</Start>
          <End>87</End>
        </Fragment>
      </Clone>
    + <Clone></Clone>
    + <Clone></Clone>
    + <Clone></Clone>
    + <Clone></Clone>
  </CloneClass>
```

Tools using GCF[1] + SimCal include

- Simian (textual report)
- iClones (RCF format)
- NiCad (XML report)
- Deckard (textual report)

[1] Wang, T., Harman, M., Jia, Y., & Krinke, J. (2013). Searching for Better Configurations: A Rigorous Approach to Clone Evaluation. FSE'13

Similarity Report (ncd-bzlib)

	InfC/ orig	InfC/ artfc	InfC/ orig no kraka tau	InfC/ orig no procy on	InfC/ orig pg kraka tau	InfC/ orig pg procy on	InfC/ artfc no kraka tau	InfC/ artfc no procy on	InfC/ artfc pg kraka tau	InfC/ artfc pg procy on	Sqrt/ orig	Sqrt/ artfc	…	Squr/ artfc pg kraka tau	Squr/ artfc pg procy on
InfConv/orig	**100**	55	36	63	32	43	34	60	31	43	20	20	…	14	17
InfConv/artifice	55	**100**	35	54	33	39	37	56	32	39	19	30	…	14	17
InfConv/orig_no_krakatau	36	35	**100**	38	60	26	80	35	59	26	13	14	…	28	17
InfConv/orig_no_procyon	63	54	38	**100**	34	58	37	80	34	58	21	20	…	15	21
InfConv/orig_pg_krakatau	32	33	60	34	**100**	33	61	33	82	33	17	17	…	29	20
InfConv/orig_pg_procyon	43	39	26	58	33	**100**	26	59	33	100	19	20	…	14	21
InfConv/artific_no_krakatau	34	37	80	37	61	26	**100**	36	59	26	14	14	…	28	17
InfConv/artifice_no_procyon	60	56	35	80	33	59	36	**100**	32	59	19	20	…	15	19
InfConv/artifice_pg_krakatau	31	32	59	34	82	33	59	32	**100**	33	15	16	…	28	17
InfConv/artifice_pg_procyon	43	39	26	58	33	100	26	59	33	**100**	19	20	…	14	21
Sqrt/orig	20	19	13	21	17	19	14	19	15	19	**100**	32	…	14	16
Sqrt/artifice	20	30	14	20	17	20	14	20	16	20	32	**100**	…	15	18
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
Square/artifice_pg_krakatau	14	14	28	15	29	14	28	15	28	14	14	15	…	**100**	32
Square/artifice_pg_procyon	17	17	17	21	20	21	17	19	17	21	16	18	…	32	**100**

ncd-bzlib with similarity threshold = 50

	InfC/ orig	InfC/ artfc	InfC/ orig no kraka tau	InfC/ orig no procy on	InfC/ orig pg kraka tau	InfC/ orig pg procy on	InfC/ artfc no kraka tau	InfC/ artfc no procy on	InfC/ artfc pg kraka tau	InfC/ artfc pg procy on	Sqrt/ orig	Sqrt/ artfc	…	Squr/ artfc pg kraka tau	Squr/ artfc pg procy on
InfConv/orig	**100**	55	36	63	32	43	34	60	31	43	20	20	…	14	17
InfConv/artifice	55	**100**	35	54	33	39	37	56	32	39	19	30	…	14	17
InfConv/orig_no_krakatau	36	35	**100**	38	60	26	80	35	59	26	13	14	…	28	17
InfConv/orig_no_procyon	63	54	38	**100**	34	58	37	80	34	58	21	20	…	15	21
InfConv/orig_pg_krakatau	32	33	60	34	**100**	33	61	33	82	33	17	17	…	29	20
InfConv/orig_pg_procyon	43	39	26	58	33	**100**	26	59	33	100	19	20	…	14	21
InfConv/artific_no_krakatau	34	37	80	37	61	26	**100**	36	59	26	14	14	…	28	17
InfConv/artifice_no_procyon	60	56	35	80	33	59	36	**100**	32	59	19	20	…	15	19
InfConv/artifice_pg_krakatau	31	32	59	34	82	33	59	32	**100**	33	15	16	…	28	17
InfConv/artifice_pg_procyon	43	39	26	58	33	100	26	59	33	**100**	19	20	…	14	21
Sqrt/orig	20	19	13	21	17	19	14	19	15	19	**100**	32	…	14	16
Sqrt/artifice	20	30	14	20	17	20	14	20	16	20	32	**100**	…	15	18
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
Square/artifice_pg_krakatau	14	14	28	15	29	14	28	15	28	14	14	15	…	**100**	32
Square/artifice_pg_procyon	17	17	17	21	20	21	17	19	17	21	16	18	…	32	**100**

ncd-bzlib with similarity threshold = 25

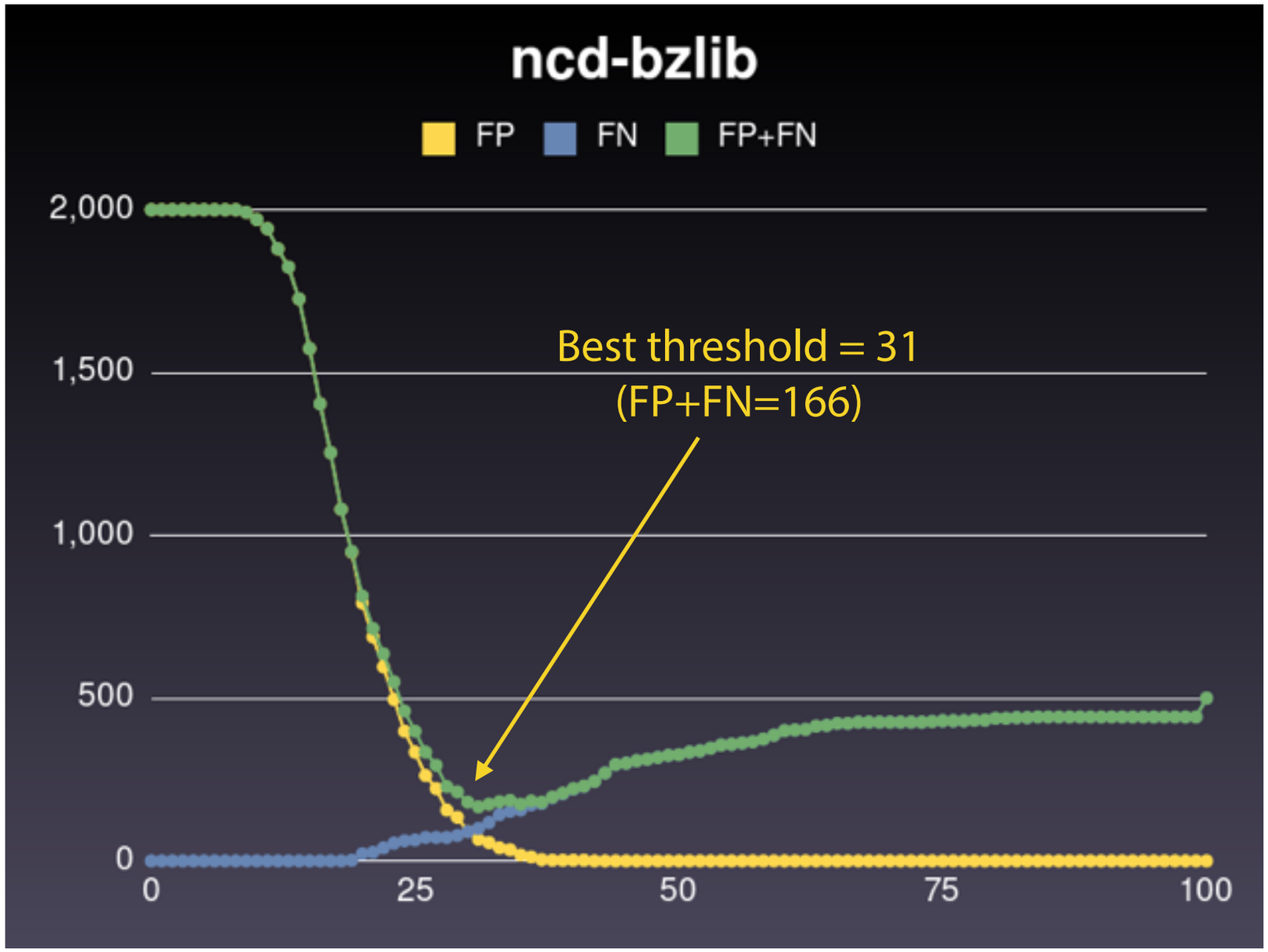
	InfC/ orig	InfC/ artfc	InfC/ orig no kraka tau	InfC/ orig no procy on	InfC/ orig pg kraka tau	InfC/ orig pg procy on	InfC/ artfc no kraka tau	InfC/ artfc no procy on	InfC/ artfc pg kraka tau	InfC/ artfc pg procy on	Sqrt/ orig	Sqrt/ artfc	…	Squr/ artfc pg kraka tau	Squr/ artfc pg procy on
InfConv/orig	**100**	55	36	63	32	43	34	60	31	43	20	20	…	14	17
InfConv/artifice	55	**100**	35	54	33	39	37	56	32	39	19	30	…	14	17
InfConv/orig_no_krakatau	36	35	**100**	38	60	26	80	35	59	26	13	14	…	28	17
InfConv/orig_no_procyon	63	54	38	**100**	34	58	37	80	34	58	21	20	…	15	21
InfConv/orig_pg_krakatau	32	33	60	34	**100**	33	61	33	82	33	17	17	…	29	20
InfConv/orig_pg_procyon	43	39	26	58	33	**100**	26	59	33	100	19	20	…	14	21
InfConv/artific_no_krakatau	34	37	80	37	61	26	**100**	36	59	26	14	14	…	28	17
InfConv/artifice_no_procyon	60	56	35	80	33	59	36	**100**	32	59	19	20	…	15	19
InfConv/artifice_pg_krakatau	31	32	59	34	82	33	59	32	**100**	33	15	16	…	28	17
InfConv/artifice_pg_procyon	43	39	26	58	33	100	26	59	33	**100**	19	20	…	14	21
Sqrt/orig	20	19	13	21	17	19	14	19	15	19	**100**	32	…	14	16
Sqrt/artifice	20	30	14	20	17	20	14	20	16	20	32	**100**	…	15	18
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
Square/artifice_pg_krakatau	14	14	28	15	29	14	28	15	28	14	14	15	…	**100**	32
Square/artifice_pg_procyon	17	17	17	21	20	21	17	19	17	21	16	18	…	32	**100**

1. Best threshold (T)

- Find the “**best threshold** (T)” of each tool with a specific parameter setting
- Calculate a sum of false positive and false negative (FP + FN) of all thresholds
- Choose T with the minimum false results

$$\mathrm{BestThreshold} = \{\mathrm{T}|\mathrm{Min}(\mathrm{FP_T} + \mathrm{FN_T})\}$$

Threshold selection



Threshold	TP	FP	TN	FN	FP+FN	Precision	Recall	F-measure (F1)
31	400	66	1934	100	166	0.8583690	0.8	0.828157

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

ncd-bzlib with similarity threshold = 31

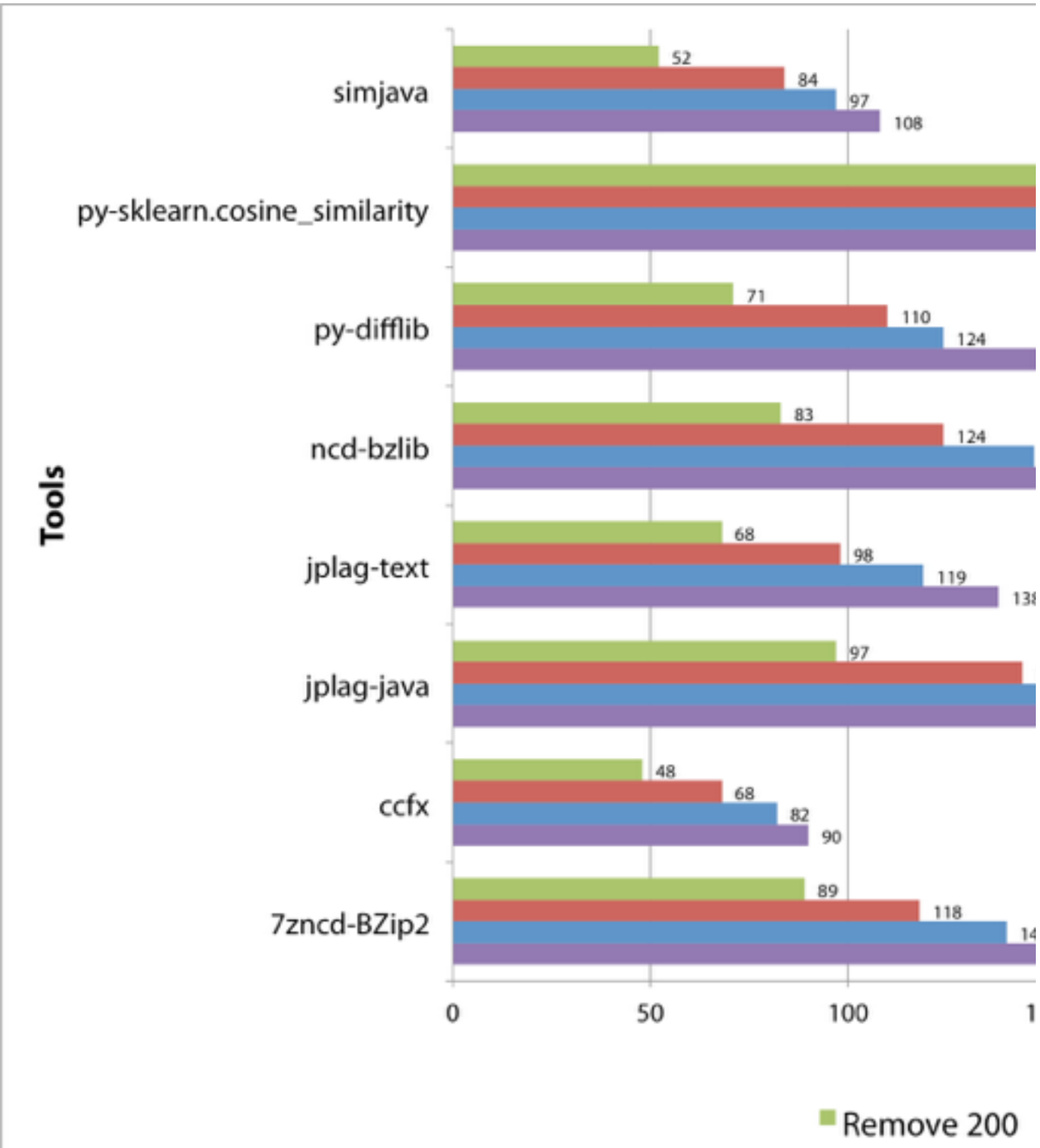
	InfC/ orig	InfC/ artfc	InfC/ orig no kraka tau	InfC/ orig no procy on	InfC/ orig pg kraka tau	InfC/ orig pg procy on	InfC/ artfc no kraka tau	InfC/ artfc no procy on	InfC/ artfc pg kraka tau	InfC/ artfc pg procy on	Sqrt/ orig	Sqrt/ artfc	…	Squr/ artfc pg kraka tau	Squr/ artfc pg procy on
InfConv/orig	**100**	55	36	63	32	43	34	60	31	43	20	20	…	14	17
InfConv/artifice	55	**100**	35	54	33	39	37	56	32	39	19	30	…	14	17
InfConv/orig_no_krakatau	36	35	**100**	38	60	26	80	35	59	26	13	14	…	28	17
InfConv/orig_no_procyon	63	54	38	**100**	34	58	37	80	34	58	21	20	…	15	21
InfConv/orig_pg_krakatau	32	33	60	34	**100**	33	61	33	82	33	17	17	…	29	20
InfConv/orig_pg_procyon	43	39	26	58	33	**100**	26	59	33	100	19	20	…	14	21
InfConv/artific_no_krakatau	34	37	80	37	61	26	**100**	36	59	26	14	14	…	28	17
InfConv/artifice_no_procyon	60	56	35	80	33	59	36	**100**	32	59	19	20	…	15	19
InfConv/artifice_pg_krakatau	31	32	59	34	82	33	59	32	**100**	33	15	16	…	28	17
InfConv/artifice_pg_procyon	43	39	26	58	33	100	26	59	33	**100**	19	20	…	14	21
Sqrt/orig	20	19	13	21	17	19	14	19	15	19	**100**	32	…	14	16
Sqrt/artifice	20	30	14	20	17	20	14	20	16	20	32	**100**	…	15	18
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
Square/artifice_pg_krakatau	14	14	28	15	29	14	28	15	28	14	14	15	…	**100**	32
Square/artifice_pg_procyon	17	17	17	21	20	21	17	19	17	21	16	18	…	32	**100**

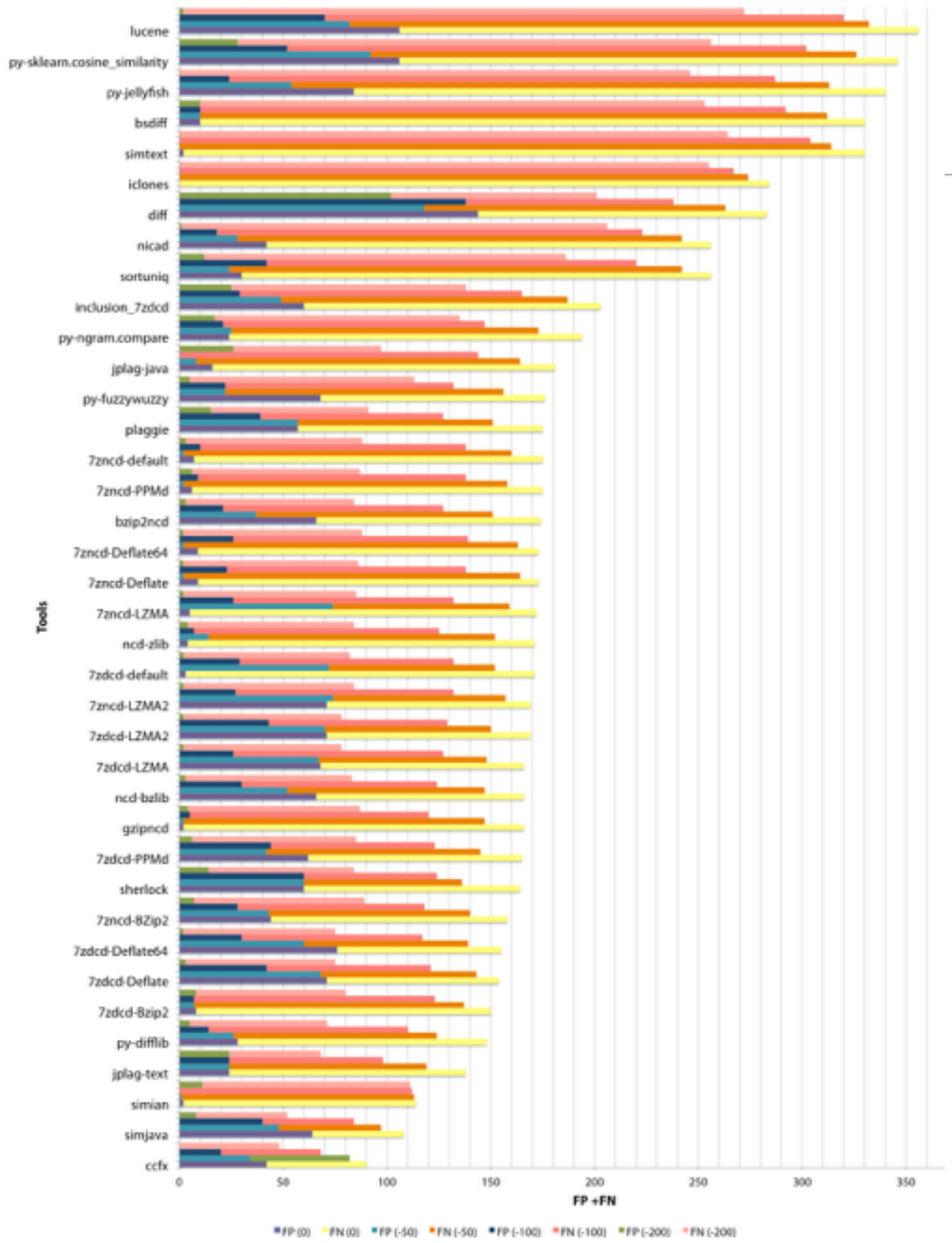
2. Manually-inspected results

- Pairs that are closest to the threshold are very sensitive to be false positive or false negative
- We have fixed cost for doing manual inspection
- Remove the top 50, 100, and 200 closest to the threshold for manual inspection
- Evaluate the new results after removal

$$\text{distance from T} = |\text{classifier(x,y)-T}|$$

RQ1: how do current detectio code obfuscation?





RQ2: what is the best parameter settings and similarity threshold of each tool?

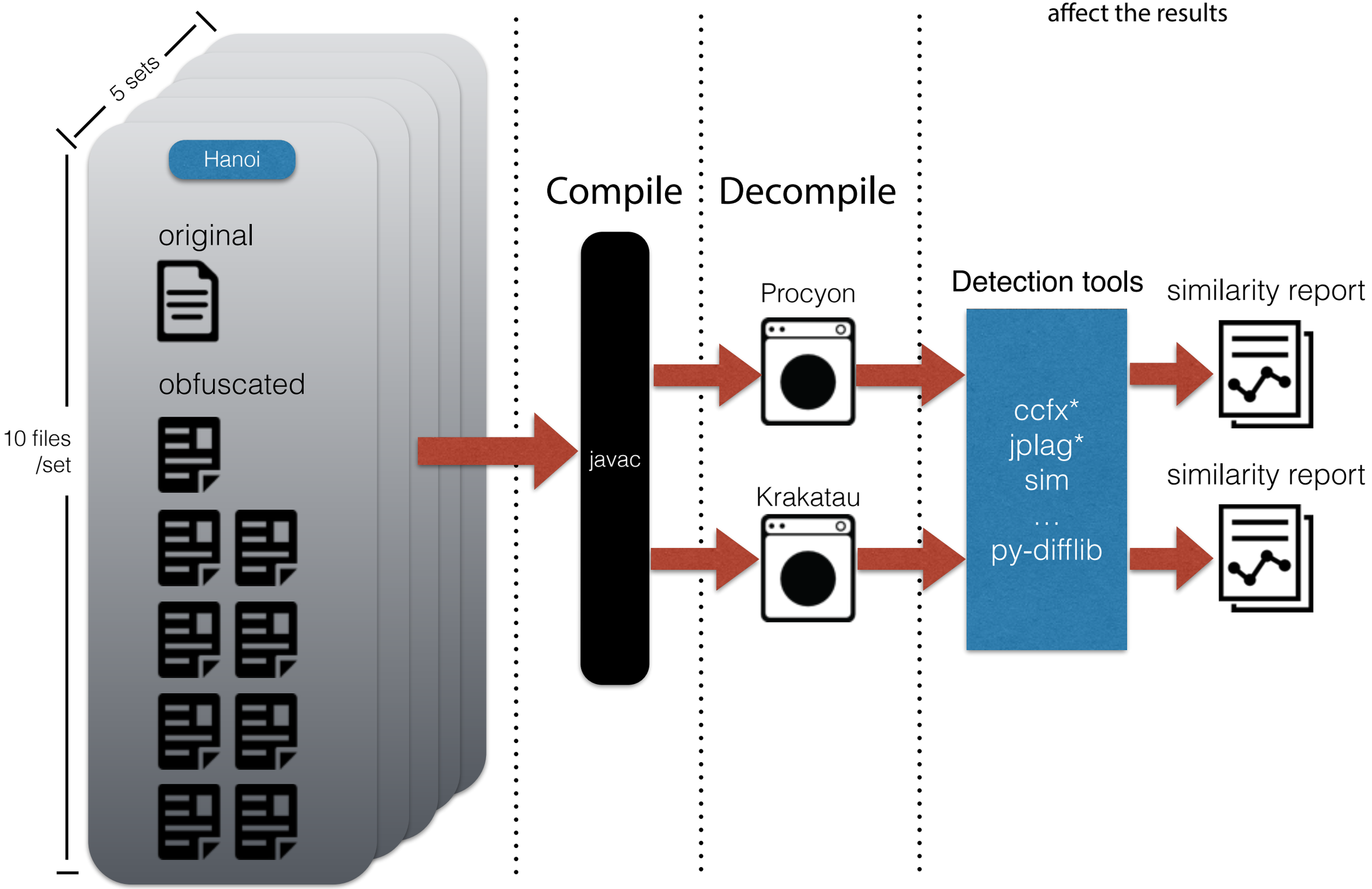
Tools	No removal			Remove 50			Remove 100			Remove 200		
	Settings	T	FP + FN	Settings	T	FP + FN	Settings	T	FP + FN	Settings	T	FP + FN
7zncd-BZip2	m0=2, mx={1,3,5}	39	158	m0=2, mx={1,3,5}	39	140	m0=2, mx={1,3,5}	39	118	m0=2, mx={1,3,5}	40	89
ncd-bzlib	-	31	166	-	32	147	-	33	124	-	34	83
ccfx[1]	b=20, t={1..7}	4	**90**	b=20, t={1..5}	4	**82**	b=20, t={1..7} b=21, t={1..7}	6	**68**	b=18, t={1..7} b=19, t={1..7} b=20, t={1..5}	8	**48**
	b=21, t={1..7}	3		b=20, t={6,7} b=21, t={1..7} b=22, t={1..7}	3					b=20, t={6,7} b=21, t={1..7} b=22, t={1..7 b=23, t={1..7}	7	
	b=22,t=7 b=23,t=7 b=24,t={1..7}	2										
jplag-java	t=3	54	181	t=7	18	164	t=6	27	144	t=3	48	97
				t=6	28		t=3	51				
jplag-text[3]	t=8	3	138	t=8	3	119	t=8	3	98	t=8	2	68
simjava[2]	r=22	5	108	r=22	6	97	r=22	6	84	r=22	10	52
py-difflib	SM_noauto junk	36	148	SM_noauto junk	36	124	SM_noauto junk	37	110	SM_ nowhitespace _noautojunk	24	71
py-sklearn.cosine_similarity	-	50	346	-	51	326	-	57	302	-	59	256

Test Case 2

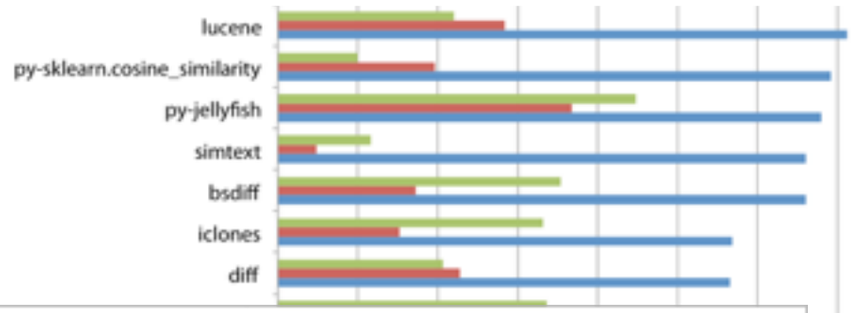
- Observation: compiling/decompiling **canonicalises** the data
- RQ3: how do compilation and decompilation facilitate the detection process?
- Experiment
- Compiled/decompiled version of the 1st dataset
- Two different decompilers: Krakatau vs Procyon
- Repeat the detection steps of Test Case 1 and compare the tool performances

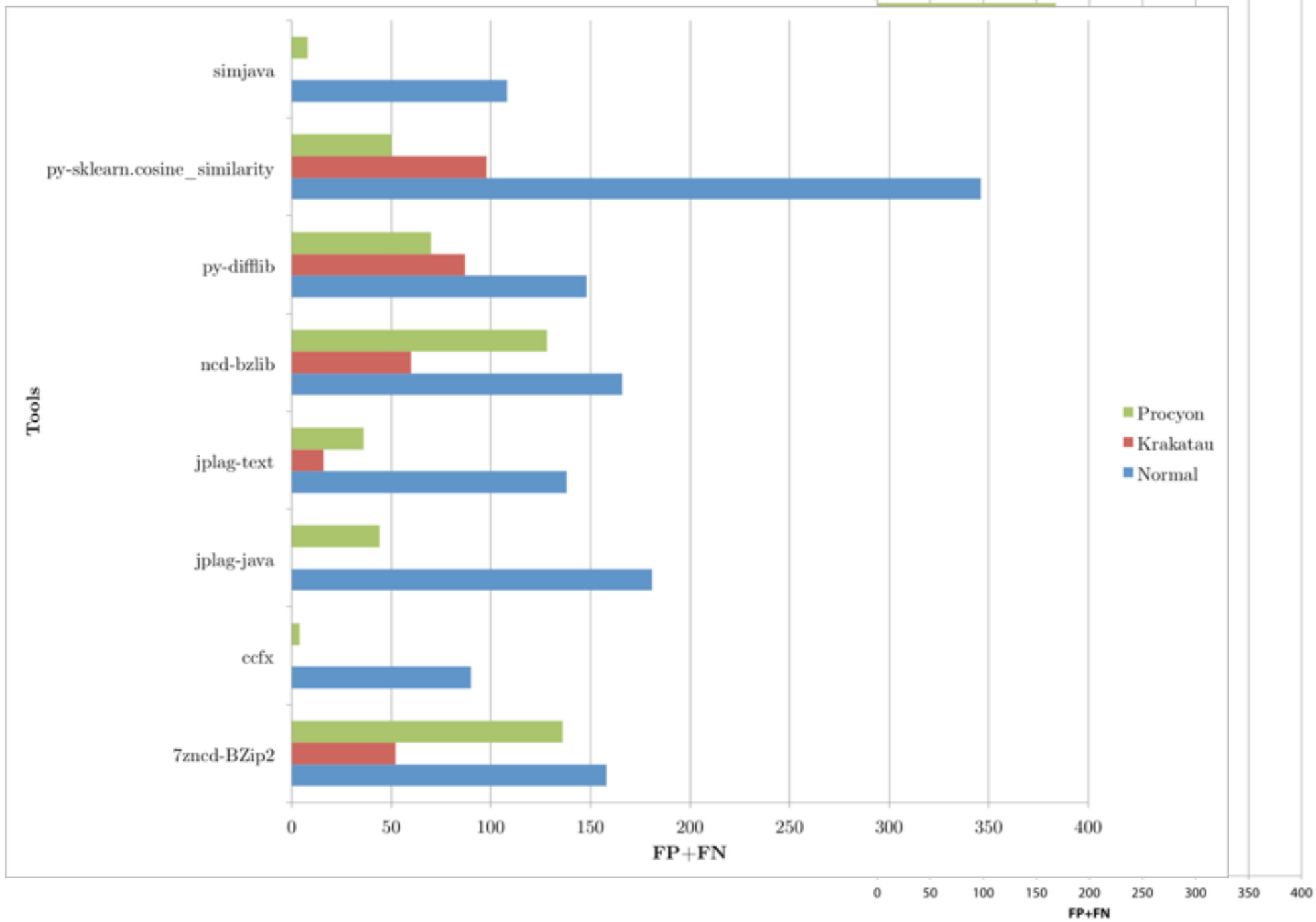
Compiling/Decompiling Process

* some tools have different parameter settings which can affect the results



RQ3: how do compilation facilitate the detection pr





Original

ARTIFICE

(source-code obfuscated version)

```java
public static String InfixToPostfixConvert ( String infixBuffer ) {
    int priority = 0;
    String postfixBuffer = "";
    Stack s1 = new Stack();
    for ( int i = 0; i < infixBuffer.length(); i++ ) {
        char ch = infixBuffer.charAt ( i );
        if ( ch == '+' || ch == '-' || ch == '*' || ch == '/' ) {
            if ( s1.size() <= 0 ) {
                s1.push ( ch );
            } else {
                Character chTop = ( Character ) s1.peek();
                if ( chTop == '*' || chTop == '/' ) {
                    priority = 1;
                } else {
                    priority = 0;
                }
                if ( priority == 1 ) {
                    if ( ch == '+' || ch == '-' ) {
                        postfixBuffer += s1.pop();
                        i--;
                    } else {
                        postfixBuffer += s1.pop();
                        i--;
                    }
                } else {
                    if ( ch == '+' || ch == '-' ) {
                        postfixBuffer += s1.pop();
                        s1.push ( ch );
                    } else {
                        s1.push ( ch );
                    }
                }
            }
        } else {
            postfixBuffer += ch;
        }
    }
    int len = s1.size();
    for ( int j = 0; j < len; j++ ) {
        postfixBuffer += s1.pop();
```

```java
public static String m20 ( String v6 ) {
    int v7;
    v7 = 0;
    String v8;
    v8 = "";
    Stack v9;
    v9 = new Stack();
    int v10;
    v10 = 0;
    while ( v10 < v6.length() ) {
        char v11;
        v11 = v6.charAt ( v10 );
        if ( v11 == '+' || v11 == '-' || v11 == '*' || v11 == '/' ) {
            if ( v9.size() <= 0 ) {
                v9.push ( v11 );
            } else {
                Character v12;
                v12 = ( Character ) v9.peek();
                if ( v12 == '*' || v12 == '/' ) {
                    v7 = 1;
                } else {
                    v7 = 0;
                }
                if ( v7 == 1 ) {
                    if ( v11 == '+' || v11 == '-' ) {
                        v8 = v8 + ( v9.pop() );
                        v10 = v10 - 1;
                    } else {
                        v8 = v8 + ( v9.pop() );
                        v10 = v10 - 1;
                    }
                } else {
                    if ( v11 == '+' || v11 == '-' ) {
                        v8 = v8 + ( v9.pop() );
                        v9.push ( v11 );
                    } else {
                        v9.push ( v11 );
                    }
                }
            }
```

Original (decompiled)

```
25  public static String InfixToPostfixConvert ( String s ) {
26      java.util.Stack a = new java.util.Stack();
27      String s0 = "";
28      int i = 0;
29      while ( i < s.length() ) {
30          String s1 = null;
31          int i0 = 0;
32          int i1 = s.charAt ( i );
33          label7: {
34              label8: {
35                  if ( i1 == 43 ) {
36                      break label8;
37                  }
38                  if ( i1 == 45 ) {
39                      break label8;
40                  }
41                  if ( i1 == 42 ) {
42                      break label8;
43                  }
44                  if ( i1 == 47 ) {
45                      break label8;
46                  }
47                  s1 = new StringBuilder().append ( s0 )
48                      .append ( ( char ) i1 ).toString();
49                  i0 = i;
50                  break label7;
51              }
52              if ( a.size() > 0 ) {
53                  int i2 = 0;
54                  String s2 = null;
55                  int i3 = 0;
56                  Character a0 = ( Character ) a.peek();
57                  int i4 = a0.charValue();
58                  label4: {
59                      label5: {
60                          label6: {
61                              if ( i4 == 42 ) {
62                                  break label6;
63                              }
```

ARTIFICE (decompiled)

```
25  public static String m20 ( String s ) {
26      java.util.Stack a = new java.util.Stack();
27      String s0 = "";
28      int i = 0;
29      while ( i < s.length() ) {
30          String s1 = null;
31          int i0 = 0;
32          int i1 = s.charAt ( i );
33          label7: {
34              label8: {
35                  if ( i1 == 43 ) {
36                      break label8;
37                  }
38                  if ( i1 == 45 ) {
39                      break label8;
40                  }
41                  if ( i1 == 42 ) {
42                      break label8;
43                  }
44                  if ( i1 == 47 ) {
45                      break label8;
46                  }
47                  s1 = new StringBuilder().append ( s0 )
48                      .append ( ( char ) i1 ).toString();
49                  i0 = i;
50                  break label7;
51              }
52              if ( a.size() > 0 ) {
53                  int i2 = 0;
54                  String s2 = null;
55                  int i3 = 0;
56                  Character a0 = ( Character ) a.peek();
57                  int i4 = a0.charValue();
58                  label4: {
59                      label5: {
60                          label6: {
61                              if ( i4 == 42 ) {
62                                  break label6;
63                              }
```

Test Case 3

- RQ4: can we apply the best parameters and threshold to other datasets effectively?
- Experiment
  - Munich dataset containing “simions”[1]
    - 109 independently developed Java programs for email address validation

[1] Juergens, E., Deissenboeck, F., & Hummel, B. (2011). Code similarities beyond copy & paste. Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, 78–87

RQ4: can we apply the best parameters and threshold to other datasets effectively?

Tools	Settings	Test Case 1 (2,500)		Test Case 3 (munich) (11,881)	
		T	FP+FN	T	FP+FN
ccfx	b=20, t={1..7}	4	90	4	4,797
simjava	r=22	5	108	5	4,680
jplag-text	t=8	3	138	3	11,770
py-difflib	SM_noauto junk	36	148	36	11,446
7zdcd-Bzip2	m0=2, mx={1,3,5}	58	150	58	11,432
ncd-bzlib	-	31	166	31	11,754
jplag-java	t=3	54	181	54	6,162
py-sklearn.cosine_similarity	-	50	346	50	10,282

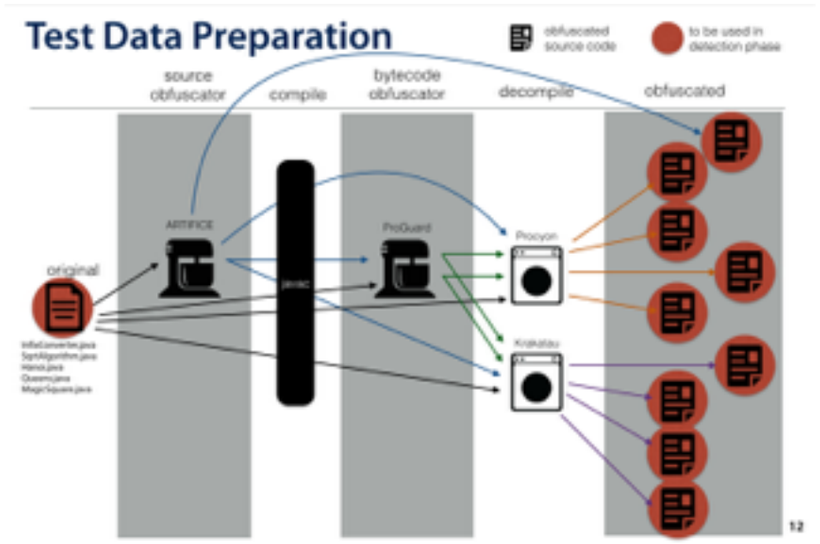
Summary

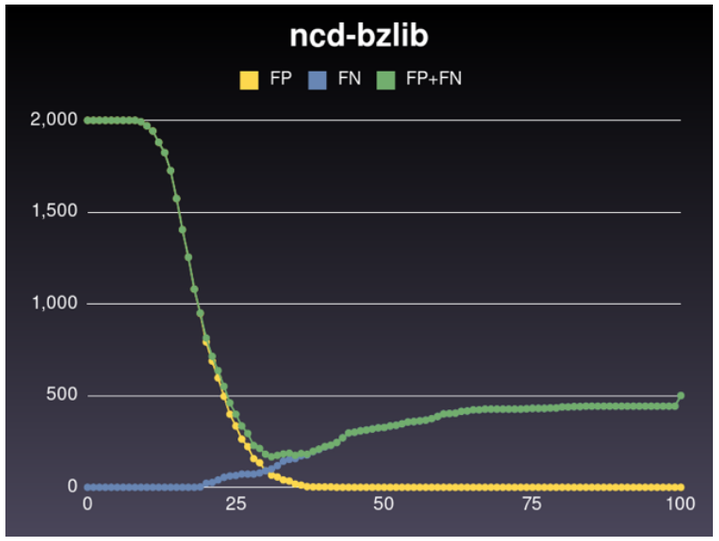
- Current tools behave differently on obfuscated code
  - Clone and plagiarism detectors outperform the others
- The best parameter settings and threshold can be found using the proposed method
- Compiling/decompiling can help canonicalise the obfuscated code
- The derived settings and threshold cannot be applied directly to other data sets

What's next?

- Replicate the experiment on other data sets and compare the best parameter settings and thresholds
- SOCO (detection of **SO**urce **CO**de re-use)
  - The Java collection contains 259 source codes
  - The C collection contains 79 source codes
- Find a better way to learn the best parameter settings and threshold in ad-hoc manner for each data set (or pair)

http://users.dsic.upv.es/grupos/nle/soco/

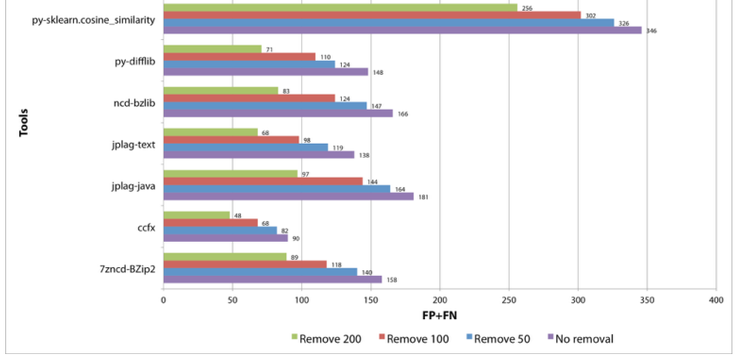




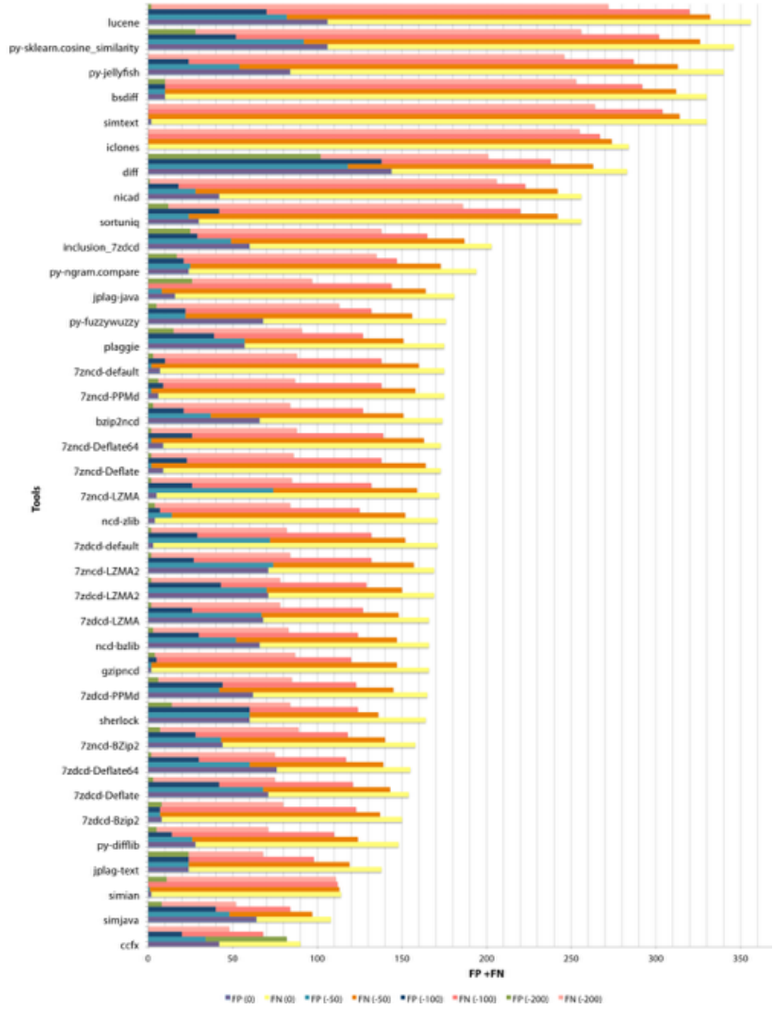
ncd-bzlib with similarity threshold = 31

20

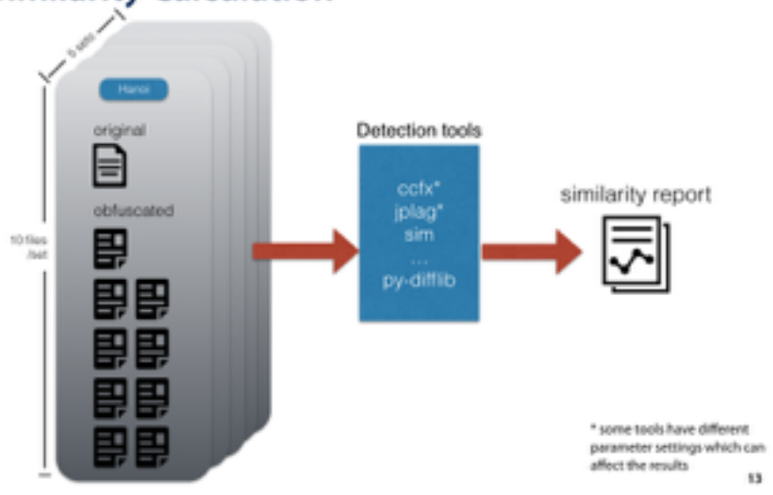
	IntCon/orig	IntCl/artific	IntCl/orig no krakatau	IntCl/orig no procyon	IntCl/orig pg krakatau	IntCl/orig pg procyon	IntCl/artific no krakatau	IntCl/artific no procyon	IntCl/artific pg krakatau	IntCl/artific pg procyon	Sqrt/orig	Sqrt/artific	...	Square/artific pg krakatau	Square/artific pg procyon
IntCon/orig	**100**	55	36	63	32	43	34	60	31	43	20	20	...	14	17
IntCon/artifice	55	**100**	35	54	33	39	37	56	32	39	19	30	...	14	17
IntCon/orig_no_krakatau	36	35	**100**	38	60	26	80	35	59	26	13	14	...	28	17
IntCon/orig_no_procyon	63	54	38	**100**	34	58	37	80	34	58	21	20	...	15	21
IntCon/orig_pg_krakatau	32	33	60	34	**100**	33	61	33	82	33	17	17	...	29	20
IntCon/orig_pg_procyon	43	39	26	58	33	**100**	26	59	33	100	19	20	...	14	21
IntCon/artific_no_krakatau	34	37	80	37	61	26	**100**	36	59	26	14	14	...	28	17
IntCon/artifice_no_procyon	60	56	35	80	33	59	36	**100**	32	59	19	20	...	15	19
IntCon/artifice_pg_krakatau	31	32	59	34	82	33	59	32	**100**	33	15	16	...	28	17
IntCon/artifice_pg_procyon	43	39	26	58	33	100	26	59	33	**100**	19	20	...	14	21
Sqrt/orig	20	19	13	21	17	19	14	19	15	19	**100**	32	...	14	16
Sqrt/artifice	20	30	14	20	17	20	14	20	16	20	32	**100**	...	15	18
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Square/artifice_pg_krakatau	14	14	28	15	29	14	28	15	28	14	14	15	...	**100**	32
Square/artifice_pg_procyon	17	17	17	21	20	21	17	19	17	21	16	18	...	32	**100**



Questions?



Similarity Calculation



Similarity Calculation for Unsupported Tools

Tools using GCF[1] + SimCal include

- Simian (textual report)
- iClones (RCF format)
- NiCad (XML report)
- Deckard (textual report)

[1] Wang, T., Harman, M., Jia, Y., & Krinke, J. (2013). Searching for Better Configurations: A Rigorous Approach to Clone Evaluation. FSE'13 14

Different Behaviours of Decompilers

Original

```
public static String InfixToPostfixConvert ( String infixBuffer ) {
    int priority = 0;
    String postfixBuffer = "";
    Stack s1 = new Stack();
    for ( int i = 0; i < infixBuffer.length(); i++ ) {
```

Krakatau

```
public static String InfixToPostfixConvert ( String s ) {
    java.util.Stack a = new java.util.Stack();
    String s0 = "";
    int i = 0;
    while ( i < s.length() ) {
```

Procyon

```
public static String InfixToPostfixConvert ( final String s ) {
    String s2 = "";
    final Stack<Character> stack = new Stack<Character>();
    for ( int i = 0; i < s.length(); ++i ) {
```

RQ3: how do compilation and decompilation facilitate the detection process?

	Test case 1		Test case 2 (Krakatau)		Test case 2 (Procyon)	
Tools	T	FP+FN	T	FP+FN	T	FP+FN
ccfx	2,3,4	90	0,5,12,15,37,44, 53,65,67	0	4,6	4
simjava	5	108	0,13,19,23,33,36, 37,38,42,46	0	11	8
jplag-text	3	138	40	16	0	36
py-difflib	36	148	28	87	22	70
7zdcd-Bzip2	58	150	60	54	59	114
ncd-bzlib	31	166	37	60	47	128
jplag-java	54	181	21,25,28,30,35, 47	0	15	44
py-sklearn.cosine_similarity	50	346	63	98	69	50

Original (dec)

```java
public static String InfixToPostfixConvert ( String s ) {
    java.util.Stack a = new java.util.Stack();
    String s0 = "";
    int i = 0;
    while ( i < s.length() ) {
        String s1 = null;
        int i0 = 0;
        int i1 = s.charAt ( i );
        label7: {
            label8: {
                if ( i1 == 43 ) {
                    break label8;
                }
                if ( i1 == 45 ) {
                    break label8;
                }
                if ( i1 == 42 ) {
                    break label8;
                }
                if ( i1 == 47 ) {
                    break label8;
                }
                s1 = new StringBuilder().append ( s0 )
                    .append ( ( char ) i1 ).toString();
                i0 = i;
                break label7;
            }
            if ( a.size() > 0 ) {
                int i2 = 0;
                String s2 = null;
                int i3 = 0;
                Character a0 = ( Character ) a.peek();
                int i4 = a0.charValue();
                label4: {
                    label5: {
                        label6: {
                            if ( i4 == 42 ) {
                                break label6;
                            }
                            int i5 = a0.charValue();
```

ARTIFICE + ProGuard (dec)

```java
private static String a ( String s ) {
    java.util.Stack a = new java.util.Stack();
    String s0 = "";
    int i = 0;
    while ( i < s.length() ) {
        String s1 = null;
        int i0 = 0;
        int i1 = s.charAt ( i );
        label0: {
            label7: {
                if ( i1 == 43 ) {
                    break label7;
                }
                if ( i1 == 45 ) {
                    break label7;
                }
                if ( i1 == 42 ) {
                    break label7;
                }
                if ( i1 == 47 ) {
                    break label7;
                }
                s1 = new StringBuilder().append ( s0 )
                    .append ( ( char ) i1 ).toString();
                i0 = i;
                break label0;
            }
            if ( a.size() > 0 ) {
                int i2 = 0;
                Character a0 = ( Character ) a.peek();
                int i3 = a0.charValue();
                label4: {
                    label5: {
                        label6: {
                            if ( i3 == 42 ) {
                                break label6;
                            }
                            int i4 = a0.charValue();
                            if ( i4 != 47 ) {
                                break label5;
```

RQ4: can we apply the best parameters and threshold to other datasets effectively?

Tools	Settings	Test Case 1 (2,500)		Test Case 3 (munich) (11,881)			
		T	FP+FN	T	FP+FN	T	FP+FN
ccfx	b=20, t={1..7}	4	90	4	4,797	82	2
simjava	r=22	5	108	5	4,680	96	2
jplag-text	t=8	3	138	3	11,770	94	3
py-difflib	SM_noauto junk	36	148	36	11,446	98	2
7zdcd-Bzip2	m0=2, mx={1,3,5}	58	150	58	11,432	85	2
ncd-bzlib	-	31	166	31	11,754	87	0
jplag-java	t=3	54	181	54	6,162	99	4
py-sklearn.cosine_similarity	-	50	346	50	10,282	99	0