CloPlag

A Study of Effects of Code Obfuscation to Clone/Plagiarism Detection Tools

Jens Krinke, Chaiyong Ragkhitwetsagul, Albert Cabré Juan

Outline

- Background
- Motivation and Research Questions
- Tools
- Experimental Design
- Results

Outline

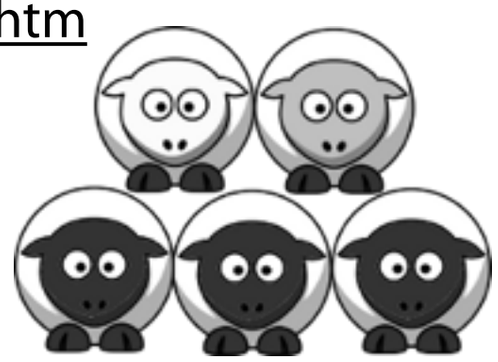
- Background
- Motivation and Research Questions
- Tools
- Experimental Design
- Results

Cloned Codes

- 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

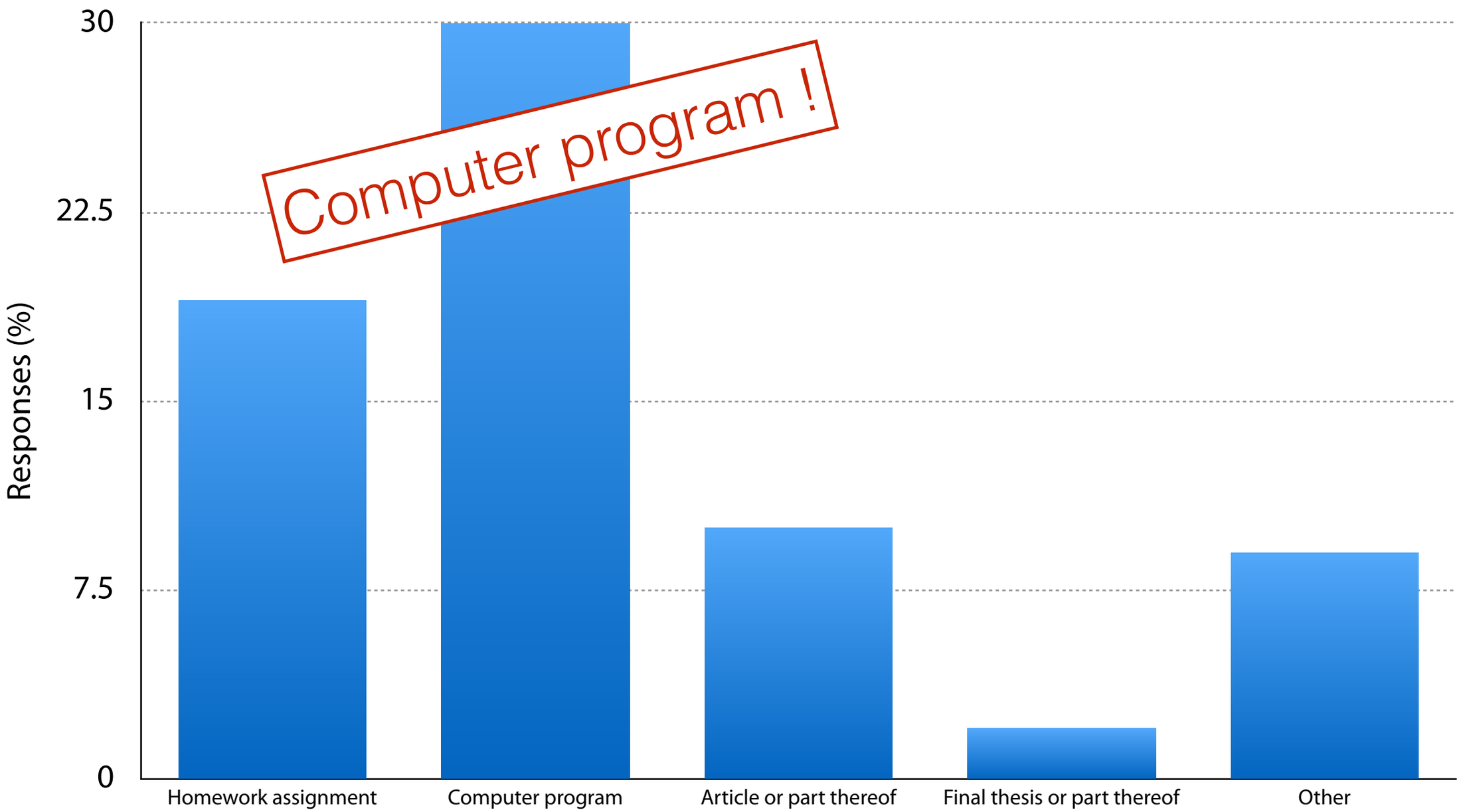
Plagiarised Codes

- Created in a similar way as code clones but with different intention
- Source code plagiarism violates academic regulations
  - CS UCL: http://www0.cs.ucl.ac.uk/teaching/staff/CS.PlagiarismGuidelines.htm
- 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/

Most Plagiarised Documents



Daniela, C., Navrat, P., Kovacova, B., & Humay, P. (2012). The Issue of (Software) Plagiarism: A Student View. IEEE Transactions on Education

Obfuscation

- Modifying program source/byte code 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
```

What is Obfuscation used for?

1. **To protect intellectual property (byte codes)**
   - Deter reverse engineering
2. **To disguise intellectual property theft (copied codes)**
   - Circumvent clone/plagiarism detection tool
3. **To disguise malware residing in a program**
   - Modify a part of code that is malware so it can avoid a virus scan program

Outline

- Background
- Motivation and Research Questions
- Tools
- Experimental Design
- Results

Motivation

- To evaluate efficiency of various types of similarity detection tool against code obfuscation
- To study effects of obfuscation at the byte-code level
- To study effects of decompilation as a type of obfuscation

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?

Outline

- Background
- Motivation and Research Questions
- Tools
- Experimental Design
- Results

Tools

Obfuscators

ARTIFICE
ProGuard

Decompilers

Procyon
Krakatau

Detection

Clone
SW plagiarism
NCD
Others

Obfuscators

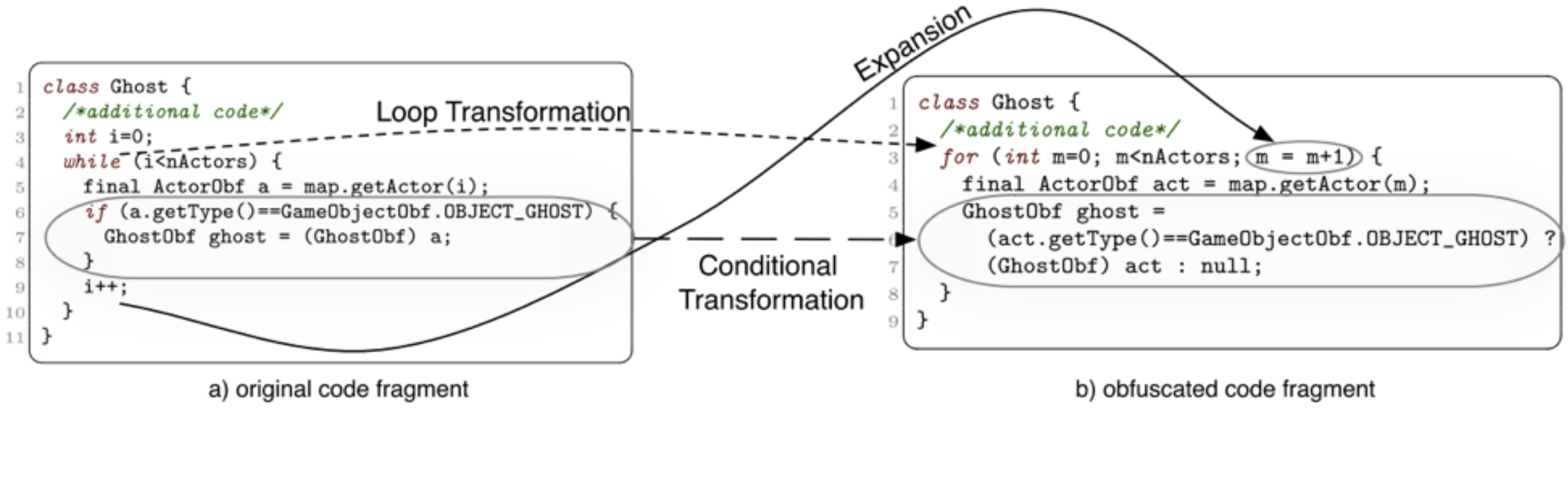
ARTIFICE

- Eclipse plug-in
- Performs the following transformations:
  - **Renaming:** rename variables, fields, and methods
  - **Expansion:** expand assignment and `++/--`
  - **Contraction:** Insert assignment and `++/—`, if possible.
  - **Loop Transformation:** change `for` to `while` loops (and vice versa)
  - **Conditional Transformation:** transform `if/else` to the equivalent short form `<E>? <A>: <B>` (and vice versa)

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

Obfuscators (cont.)

Example of ARTIFICE transformations



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

Obfuscator (cont.)

ProGuard

- A free Java class file shrinker, optimiser, obfuscator, and preverifier.
- Operate at bytecode level
  - **Shrinker:** detects and removes unused classes, fields, methods, and attributes.
  - **Optimiser:** optimises bytecode and removes unused instructions.
  - **Obfuscator:** renames the remaining classes, fields, and methods using short meaningless names.
  - **Preverifier:** preverifies the processed code for Java 6 or higher, or for Java Micro Edition.

http://proguard.sourceforge.net

Detection Tools

Clone detection

CCFinderX

ncd-bzlib

7zncd-BZip2

NCD

Source code plagiarism detection

JPlag

Sim

py-difflib

py-sklearn.cosine_similarity

Others

* All tools report similarity values (0 - 100)

Outline

- Background
- Motivation and Research Questions
- Tools
- Experimental Design
- Results

Data Set

A series of Java programs

- InfixConverter
- SqrtAlgorithm
- Hanoi
- Queens
- MagicSquare

```
public static String InfixToPostfixConvert ( final String s ) {
    String s2 = "";
    final Stack<Character> stack = new Stack<Character>();
    for ( int i = 0; i < s.length(); ++i ) {
        final char char1 = s.charAt ( i );
        if ( char1 == '+' || char1 == '-' || char1 == '*' || char1 == '/' ) {
            if ( stack.size() <= 0 ) {
                stack.push ( char1 );
            } else {
                final Character c = stack.peek();
                if ( c == '*' || c == '/' ) {
                    if ( char1 == '+' || char1 == '-' ) {
                        s2 += stack.pop();
                        --i;
                    } else {
                        s2 += stack.pop();
                        --i;
                    }
                } else if ( char1 == '+' || char1 == '-' ) {
                    s2 += stack.pop();
                    stack.push ( char1 );
                } else {
                    stack.push ( char1 );
                }
            }
        } else {
            s2 += char1;
        }
    }
    for ( int size = stack.size(), j = 0; j < size; ++j ) {
        s2 += stack.pop();
    }
    return s2;
}
```

Test Data Preparation

obfuscated source code

to be used in detection phase

source obfuscator

compile

bytecode obfuscator

decompile

obfuscated

ARTIFICE

ProGuard

Procyon

Krakatau

javac

original

InfixConverter.java
SqrtAlgorithm.java
Hanoi.java
Queens.java
MagicSquare.java

Original vs ARTIFICE + ProGuard + Krakatau

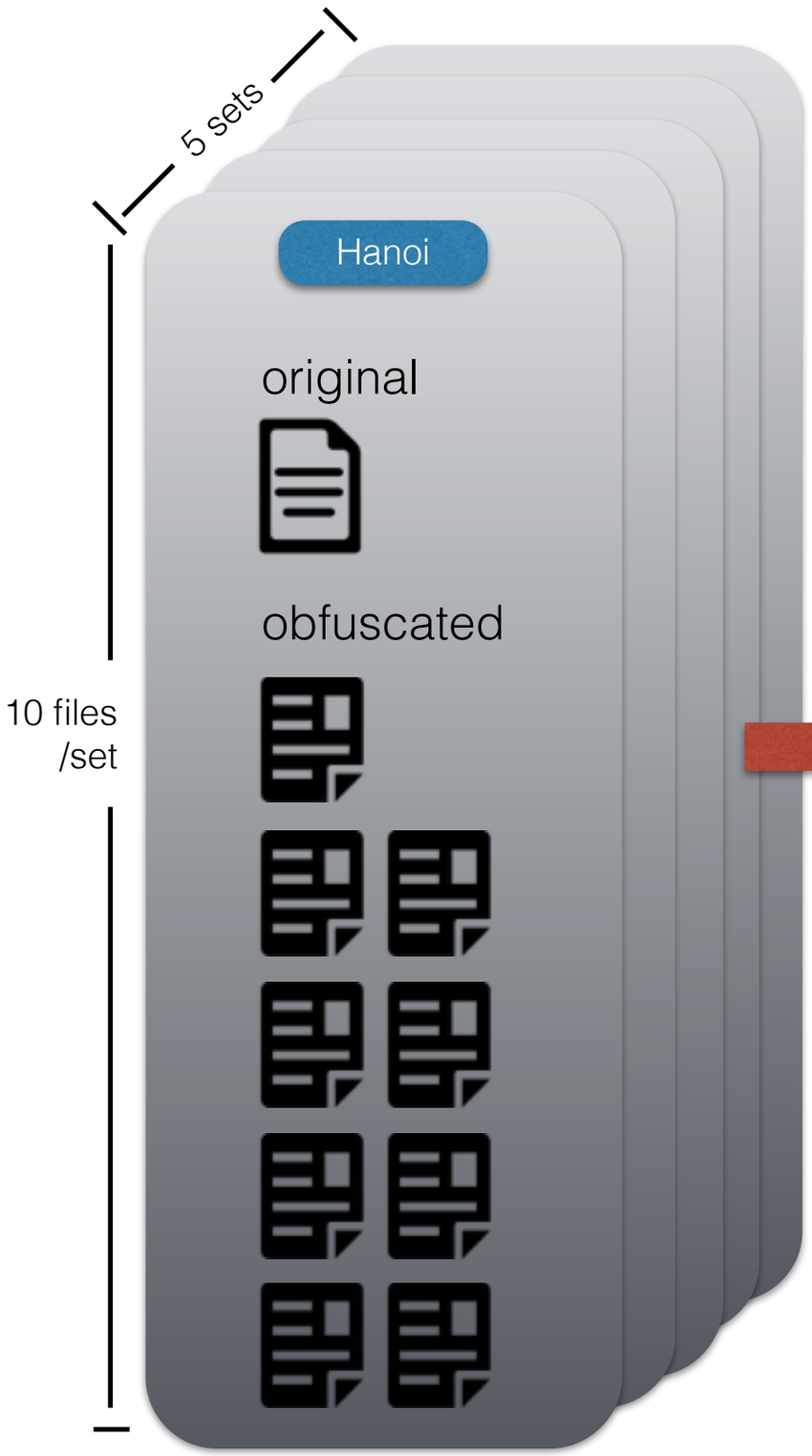
```
105     public static void main ( String[] args ) {
106         try {
107             while ( true ) {
108                 System.out.print ( "\nEnter the number of discs (-1 to exit):
    " );
109                 int maxdisc = 0;
110                 String inpstring = "";
111                 InputStreamReader input = new InputStreamReader ( System.in );
112                 BufferedReader reader = new BufferedReader ( input );
113                 inpstring = reader.readLine();
114                 movecount = 0;
115                 maxdisc = Integer.parseInt ( inpstring );
116                 if ( maxdisc == -1 ) {
117                     System.out.println ( "Good Bye!" );
118                     return;
119                 }
120                 if ( maxdisc <= 1 || maxdisc >= 10 ) {
121                     System.out.println ( "Enter between 2 - 9" );
122                     continue;
123                 }
124                 for ( int i = maxdisc; i >= 1; i-- ) {
125                     A.push ( i );
126                 }
127                 countA = A.size();
128                 countB = B.size();
129                 countC = C.size();
130                 PrintStacks();
131                 SolveTOH ( maxdisc, A, B, C );
132                 System.out.println ( "Total Moves = " + movecount );
133                 while ( C.size() > 0 ) {
134                     C.pop();
135                 }
136             }
137         } catch ( Exception e ) {
138             e.printStackTrace();
139         }
140     }
141 }
142
```

hanoi/Main.java

```
114     public static void main ( String[] a0 ) {
115         try {
116             while ( true ) {
117                 System.out.print ( "\nEnter the number of discs (-1 to exit):
    " );
118                 String s = new java.io.BufferedReader ( ( java.io.Reader ) new
    java.io.InputStreamReader ( System.in ) ).readLine();
119                 a = 0;
120                 int i = Integer.parseInt ( s );
121                 if ( i != -1 ) {
122                     if ( i > 1 && i < 10 ) {
123                         int i0 = i;
124                         while ( i0 > 0 ) {
125                             b.push ( ( Object ) Integer.valueOf ( i0 ) );
126                             i0 = i0 + -1;
127                         }
128                         e = b.size();
129                         f = c.size();
130                         g = d.size();
131                         Main.a();
132                         Main.a ( i, b, c, d );
133                         System.out.println ( new StringBuilder ( "Total Moves
    = " ).append ( a ).toString() );
134                         while ( d.size() > 0 ) {
135                             d.pop();
136                         }
137                         continue;
138                     }
139                     System.out.println ( "Enter between 2 - 9" );
140                 } else {
141                     break;
142                 }
143             }
144             System.out.println ( "Good Bye!" );
145         } catch ( Exception a1 ) {
146             a1.printStackTrace();
147             return;
148         }
149     }
```

hanoi/Main.java
(obfuscated + decompiled)

Similarity Detection



Detection tools

ccfx*
jplag*
sim
…
py-difflib

similarity report

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

Outline

- Background
- Motivation and Research questions
- Tools
- Experimental design
- Results

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

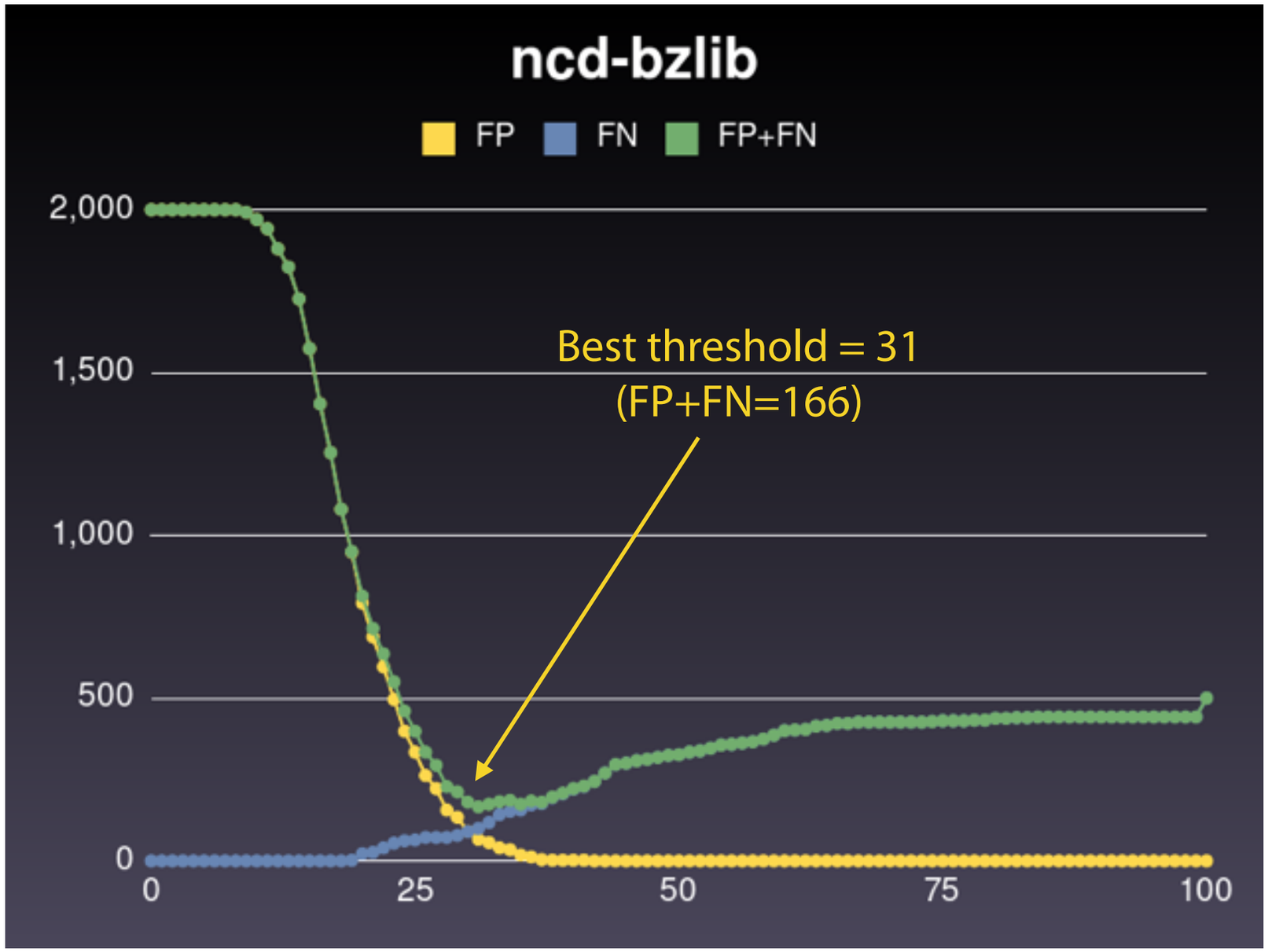
	InfC/ orig	InfC/ artfc	InfC/ orig no kraka tau	InfC/ orig no procy on	InfC/ orig pg kraka tau	InfC/ orig pg procy on	InfC/ artfc no kraka tau	InfC/ artfc no procy on	InfC/ artfc pg kraka tau	InfC/ artfc pg procy on	Sqrt/ orig	Sqrt/ artfc	…	Squr/ artfc pg kraka tau	Squr/ artfc pg procy on
InfConv/orig	**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)

- Determine the **“best threshold (T)”** for each specific tool and for each of its specific parameter setting, if any
- Calculate a sum between False Positive and False Negative (FP + FN)

$$\text{Best Threshold} = \text{MIN}(\text{FP} + \text{FN})$$

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

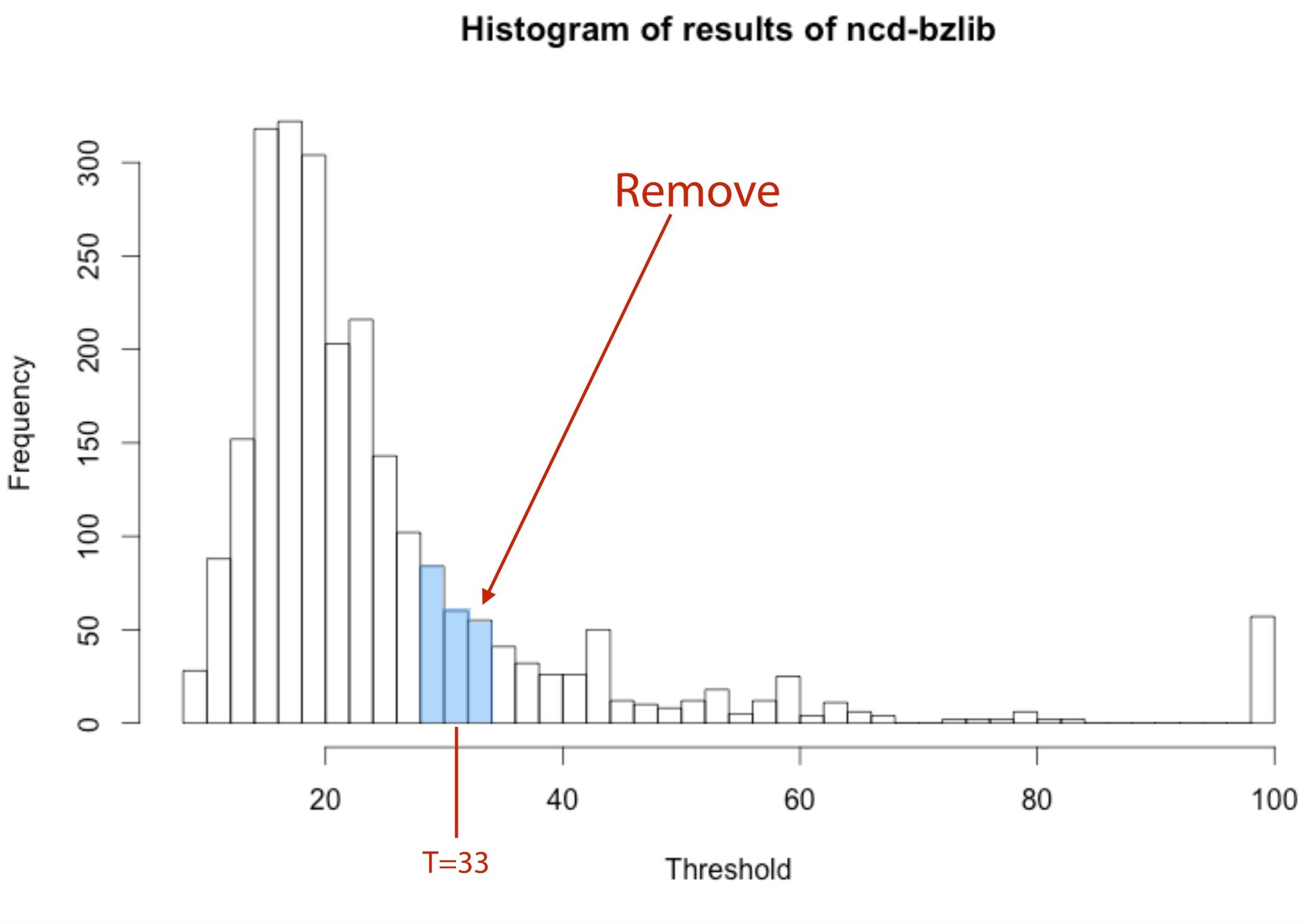
$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

2. Manually-inspected results

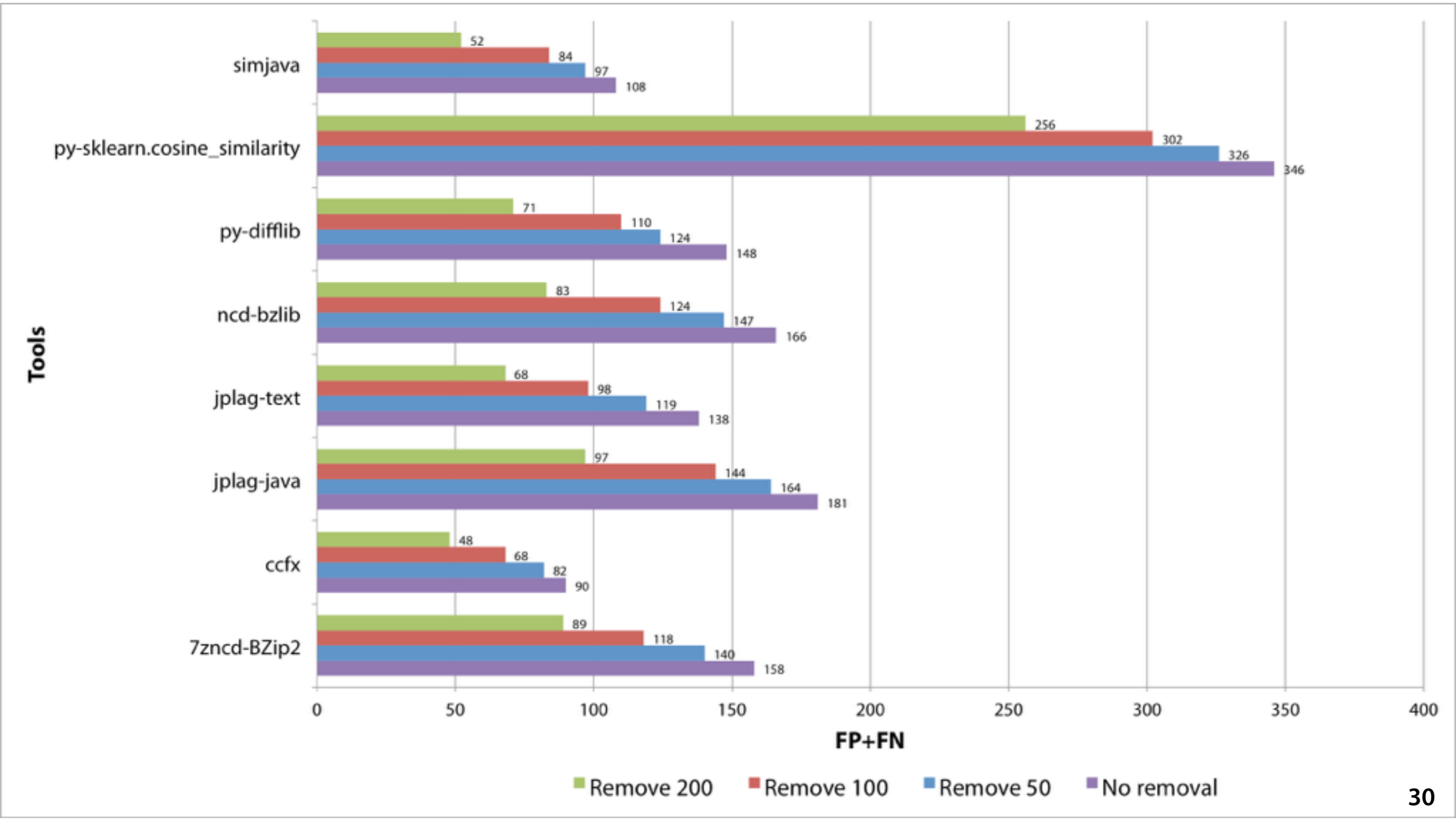
- Pairs that are closest to the threshold are very possible to be false positive or false negative
- We have fixed cost for doing manual inspection
- Remove the top 50, 100, and 200 closest to the threshold for manual inspection
- Evaluate the new results after removal

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

2. Manually-inspected results (cont.)



RQ1: how do current detection tools perform against 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={2..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

What's next?

- Run the tools with the derived best parameters settings and best thresholds against different data set
- SOCO (detection of SOurce COde re-use)
  - The Java collection contains 259 source codes
  - The C collection contains 79 source codes
- Replicate the experiment on SOCO data and compare the best parameter settings and thresholds

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

Questions?

1. Best threshold (cont.)

- Since the number of positive and negative results are in different magnitude

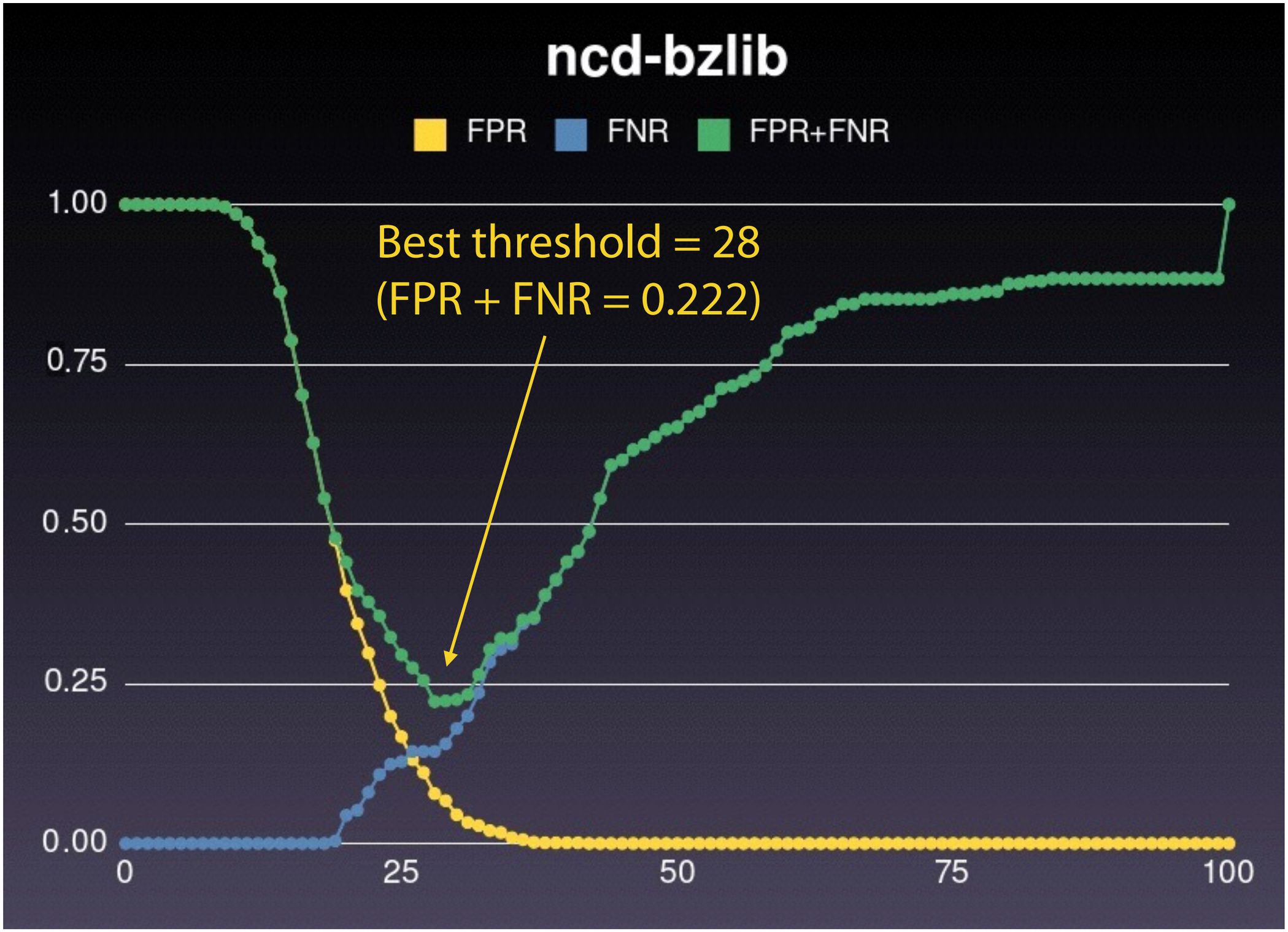
Positive	Negative
500	2000

- We also tried determine the best threshold by using a sum of False Positive Rate and False Negative Rate (FPR + FNR)

$$\text{Best Threshold} = \text{MIN}(\text{FPR} + \text{FNR})$$

$$\text{FPR} = \text{FP}/(\text{FP}+\text{TN})$$
$$\text{FNR} = \text{FN}/(\text{FN}+\text{TP})$$

Threshold selection (cont.)



Threshold Selection (cont.)

Best threshold for ncd-bzlib using FPR + FNR

Threshold	FP	FN	FP+FN	FPR	FNR	FPR+FNR	Precision	Recall	F-measure (F1)
28	156	72	228	0.078	0.144	0.222	0.733	0.856	0.7897
33	66	100	166	0.033	0.200	0.233	0.859	0.800	0.8282