Entropy-Based Test Generation for Improved Fault Localization

José Campos
jose@computer.org
University of Porto
Portugal

Rui Abreu
rui@computer.org
University of Porto
Portugal

Gordon Fraser
gordon.fraser@sheffield.ac.uk
University of Sheffield
England, UK

Marcelo d'Amorim
damorim@cin.ufpe.br
Federal University of Pernambuco
Recife, Brazil

November 14th, 2013
28th IEEE/ACM International Conference on Automated Software Engineering (ASE)
Silicon Valley, California, USA

Entropy-Based Test Generation for Improved **Fault Localization**

Entropy-Based Test Generation for Improved Fault Localization

Entropy-Based **Test Generation** for Improved Fault Localization

Entropy-Based Test Generation for Improved Fault Localization

Program

Test Suite

class Triangle {… static int type(int a, int b, int c) {	t_1	t_2	t_3	t_4	t_5	t_6	Suspiciousness
int type = SCALENE;	●	●	●	●			0.09998
if ((a == b) && (b == c))	●	●	●	●			0.09998
type = EQUILATERAL;	●						0.10001
else if ((a*a) == ((b*b) + (c*c)))		●	●	●			0.09999
type = RIGHT;			●				0.10001
else if ((a == b) || (b == a)) /* FAULT */		●		●			**0.10000**
type = ISOSCELES;		●					0.10001
return type; }	●	●	●	●			0.09998
static double area(int a, int b, int c) {							
double s = (a+b+c)/2.0;					●	●	0.10000
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }					●	●	0.10000

Fault

Spectra

SFL, A Popular Automated Approach to Assist Programmers in Debugging

Empirical Evaluation of the Tarantula Automatic Fault-Localization Technique

ASE '05

James A. Jones and Mary Jean Harrold
College of Computing, Georgia Institute of Technology
Atlanta, Georgia, U.S.A.
jjones@cc.gatech.edu, harrold@cc.gatech.edu

440 citations

On the Accuracy of Spectrum-based Fault Localization*

TAICPART '07

Rui Abreu Peter Zoeteweij Arjan J.C. van Gemund
Software Technology Department
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology
P.O. Box 5031, NL-2600 GA Delft, The Netherlands
{r.f.abreu, p.zoeteweij, a.j.c.vangemund}@tudelft.nl

201 citations

An Empirical Study of the Effects of Test-Suite Reduction on Fault Localization

ICSE '08

Yanbing Yu yyu@cc.gatech.edu
James A. Jones jjones@cc.gatech.edu
Mary Jean Harrold harrold@cc.gatech.edu
College of Computing
Georgia Institute of Technology
Atlanta, GA, U.S.A.

114 citations

Lightweight Fault-Localization Using Multiple Coverage Types

ICSE '09

Raul Santelices,† James A. Jones,‡ Yanbing Yu,† and Mary Jean Harrold†
†College of Computing, Georgia Institute of Technology
‡Department of Informatics, University of California, Irvine
†{raul|yyu|harrold}@cc.gatech.edu, ‡jajones@ics.uci.edu

103 citations

A family of code coverage-based heuristics for effective fault localization ☆

JSS '10

W. Eric Wong [a,*], Vidroha Debroy [a], Byoungju Choi [b]
[a] Department of Computer Science, University of Texas at Dallas, TX 75083, USA
[b] Department of Computer Science and Engineering, Ewha womans University, Republic of Korea

67 citations

On the Influence of Multiple Faults on Coverage-Based Fault Localization

ISSTA '11

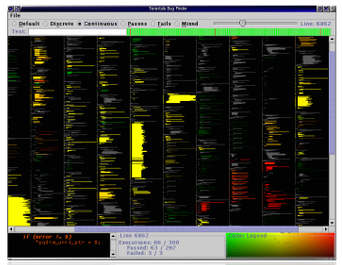
Nicholas DiGiuseppe
University of California, Irvine
Department of Informatics
ndigiuse@ics.uci.edu
James A. Jones
University of California, Irvine
Department of Informatics
jajones@ics.uci.edu

18 citations

SFL, A Popular Automated Approach to Assist Programmers in Debugging

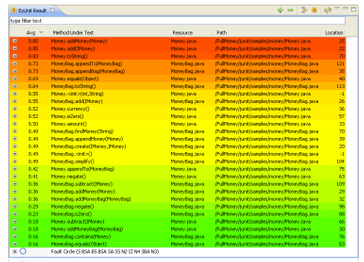
TARANTULA

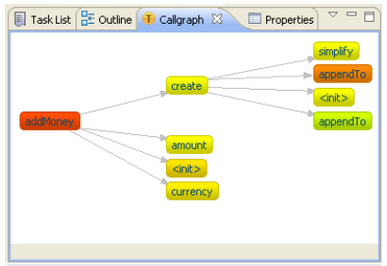
Fault Localization via Visualization



http://pleuma.cc.gatech.edu/aristotle/Tools/tarantula/

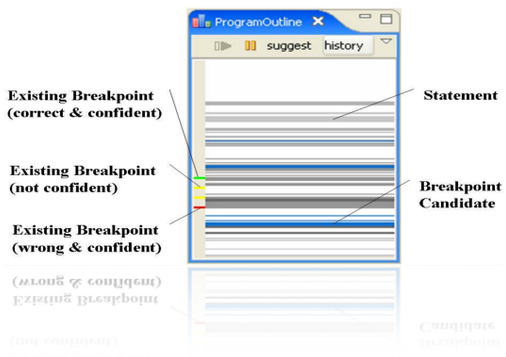
EzUnit





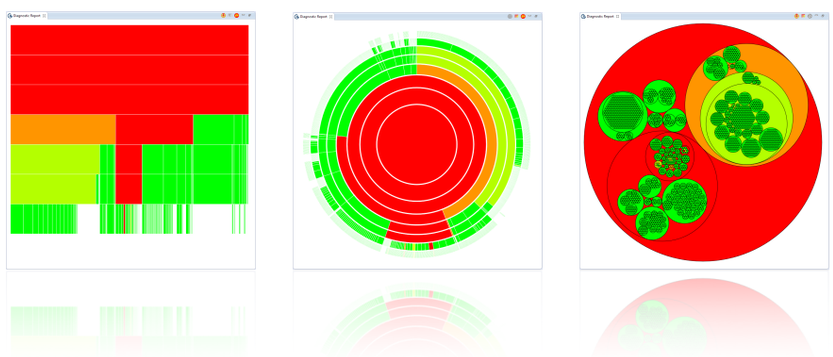
http://www.fernuni-hagen.de/ps/prjs/EzUnit4/

VIDA



Dan Hao, Lingming Zhang, Lu Zhang, Jiasu Sun, Hong Mei

GZOLTAR



http://www.gzoltar.com/

Are Automated Debugging Techniques Actually Helping Programmers?

ISSTA '11

Chris Parnin and Alessandro Orso
Georgia Institute of Technology
College of Computing
{chris.parnin|orso}@gatech.edu

ABSTRACT

Debugging is notoriously difficult and extremely time consuming. Researchers have therefore invested a considerable amount of effort in developing automated techniques and tools for supporting various debugging tasks. Although potentially useful, most of these techniques have yet to demonstrate their practical effectiveness. One common limitation of existing approaches, for instance, is their reliance on a set of strong assumptions on how developers behave when debugging (*e.g.*, the fact that examining a faulty statement in isolation is enough for a developer to understand and fix the corresponding bug). In more general terms, most existing techniques just focus on selecting subsets of potentially faulty statements and ranking them according to some criterion. By doing so, they ignore the fact that understanding the root cause of a failure typically involves complex activities, such as navigating program dependencies and rerunning the program with different inputs. The overall goal of this research is to investigate how developers use and bene-

second activity, *fault understanding*, involves understanding the root cause of the failure. Finally, *fault correction* is determining how to modify the code to remove such root cause. Fault localization, understanding, and correction are referred to collectively with the term *debugging*.

Debugging is often a frustrating and time-consuming experience that can be responsible for a significant part of the cost of software maintenance [25]. This is especially true for today's software, whose complexity, configurability, portability, and dynamism exacerbate debugging challenges. For this reason, the idea of reducing the costs of debugging tasks through techniques that can improve efficiency and effectiveness of such tasks is ever compelling. In fact, in the last few years, there has been a great number of research techniques that support automating or semi-automating several debugging activities (*e.g.*, [1,3,8,11,21,29–31]). Collectively, these techniques have pushed forward the state of the art in debugging. However, there are several challenges in scaling and transitioning these techniques that must be addressed

So, is SFL a dead-end avenue of research?

class Triangle {… static int type(int a, int b, int c) {	t_1	t_2	t_3	t_4	t_5	t_6	Suspiciousness
int type = SCALENE;	●	●	●	●			0.09998
if ((a == b) && (b == c))	●	●	●	●			0.09998
type = EQUILATERAL;	●						0.10001
else if ((a*a) == ((b*b) + (c*c)))		●	●	●			0.09999
type = RIGHT;			●				0.10001
else if ((a == b) || (b == a)) /* FAULT */		●		●			**0.10000**
type = ISOSCELES;		●					0.10001
return type; }	●	●	●	●			0.09998
static double area(int a, int b, int c) {							
double s = (a+b+c)/2.0;					●	●	0.10000
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }					●	●	0.10000

class Triangle {… static int type(int a, int b, int c) {	t_1	t_2	t_3	t_4	t_5	t_6	Suspiciousness
int type = SCALENE;	●	●	●	●			0.09998
if ((a == b) && (b == c))	●	●	●	●			0.09998
type = EQUILATERAL;							0.10001
else if ((a*a) == ((b*b) + (c*c)))		●					0.09999
type = RIGHT;			●				0.10000
else if ((a == b) || (b == a)) /* FAULT */				●			0.10000
type = ISOSCELES;							0.10001
return type; }	●	●	●	●			0.09998
static double area(int a, int b, int c) {							
double s = (a+b+c)/2.0;					●	●	0.10000
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } … }					●	●	0.10000

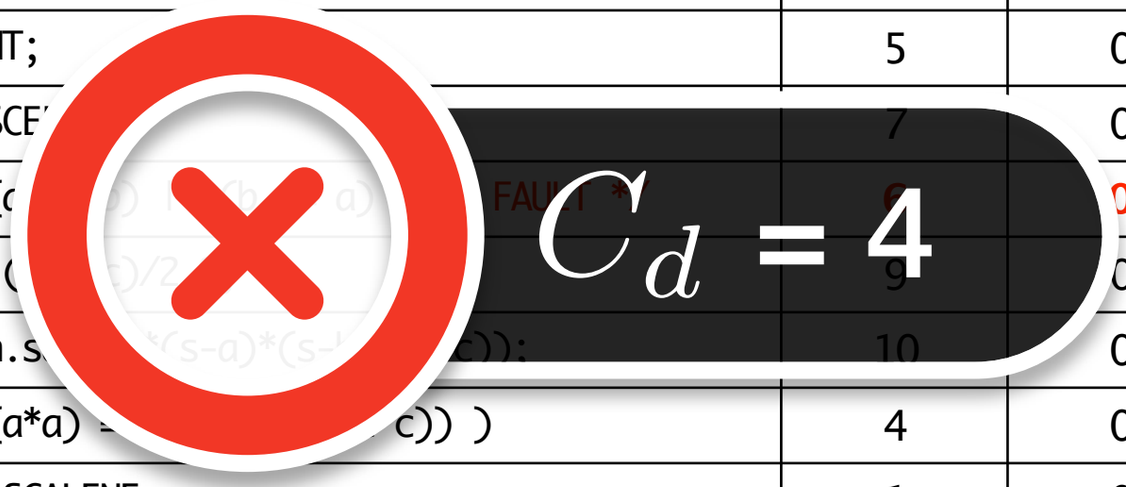
“The main confounding factor for the usefulness of SFL is the dependency on the **quality of the existing test suite**”

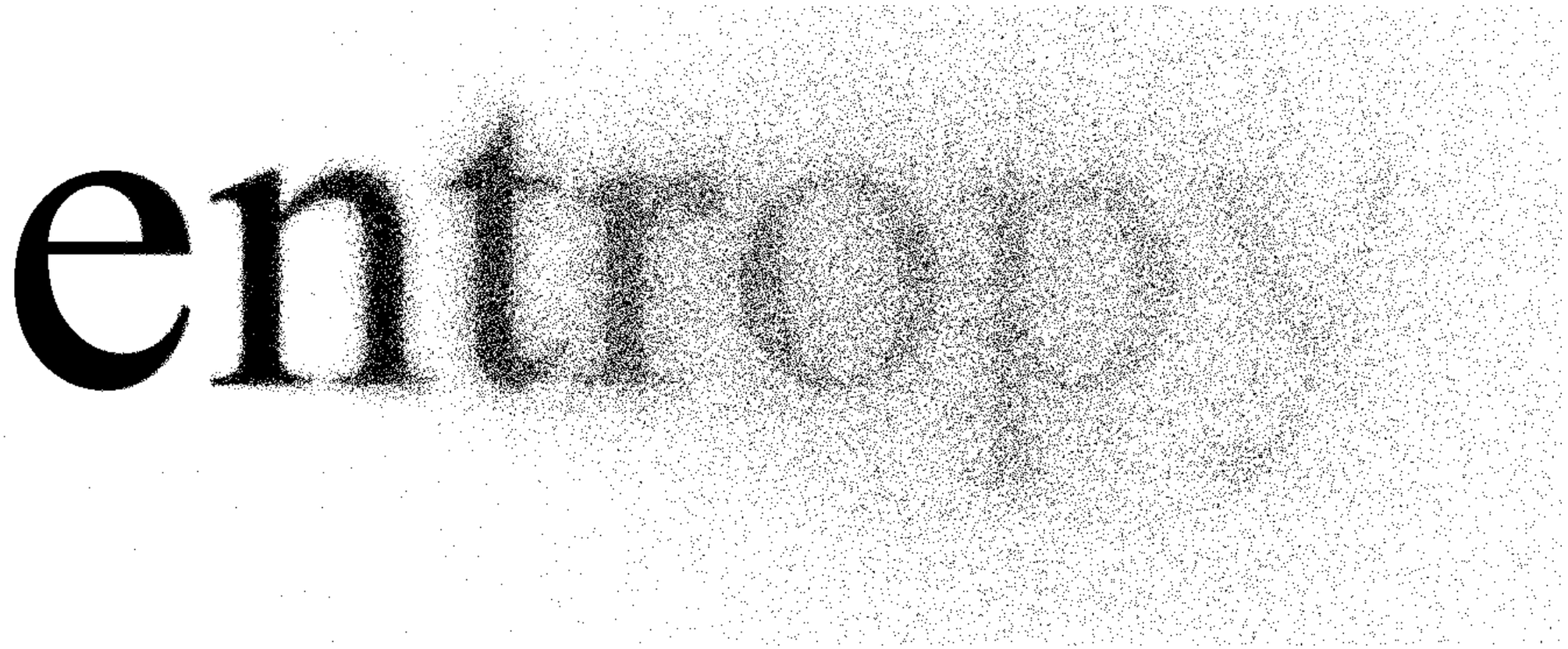
DIAGNOSTIC QUALITY

Rank Position	Suspicious Statement	Line number	Suspiciousness
1º	`type = EQUILATERAL;`	3	0.10001
2º	`type = RIGHT;`	5	0.10001
3º	`type = ISOSCELES;`	7	0.10001
4º	`else if ( (a == b) || (b == a) ) /* FAULT */`	**6**	**0.10000**
5º	`double s = (a+b+c)/2.0;`	9	0.10000
6º	`return Math.sqrt(s*(s-a)*(s-b)*(s-c));`	10	0.10000
7º	`else if ( (a*a) == ((b*b) + (c*c)) )`	4	0.09999
8º	`int type = SCALENE;`	1	0.09998
9º	`if ( (a == b) && (b == c) )`	2	0.09998
10º	`return type; }`	8	0.09998

Diagnostic Quality

Rank Position	Suspicious Statement	Line number	Suspiciousness
1º	type = EQUILATERAL;	3	0.10001
2º	type = RIGHT;	5	0.10001
3º	type = ISOSCE	7	0.10001
4º	else if ((c	6	0.10000
5º	double s = (	9	0.10000
6º	return Math.s	10	0.10000
7º	else if ((a*a) = c)))	4	0.09999
8º	int type = SCALENE;	1	0.09998
9º	if ((a == b) && (b == c))	2	0.09998
10º	return type; }	8	0.09998





$$\mathcal{H}(D) = -\sum_{d_k \in D} \Pr(d_k) \cdot \log_2(\Pr(d_k)),\ 0 \leq \mathcal{H} \leq \log_2(M)$$

A. Gonzalez-Sanchez, R. Abreu, H.-G. Gross, and A. J. van Gemund, "Spectrum-Based Sequential Diagnosis", AAAI '11

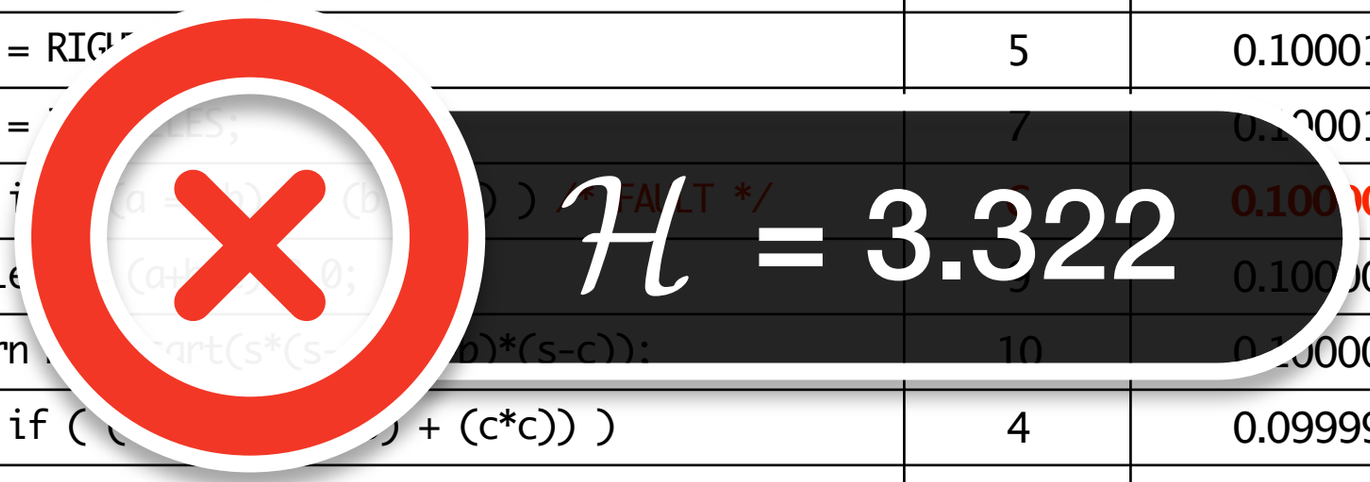
http://images3.wikia.nocookie.net/__cb20130813113160/creepypasta/images/8/8b/1entropy.gif

MEASURING ENTROPY

Rank Position	Suspicious Statement	Line number	Suspiciousness
1º	type = EQUILATERAL;	3	0.10001
2º	type = RIGHT;	5	0.10001
3º	type = ISOSCELES;	7	0.10001
4º	else if ((a == b) || (b == a)) /* FAULT */	**6**	**0.10000**
5º	double s = (a+b+c)/2.0;	9	0.10000
6º	return Math.sqrt(s*(s-a)*(s-b)*(s-c));	10	0.10000
7º	else if ((a*a) == ((b*b) + (c*c)))	4	0.09999
8º	int type = SCALENE;	1	0.09998
9º	if ((a == b) && (b == c))	2	0.09998
10º	return type; }	8	0.09998

Measuring Entropy

Rank Position	Suspicious Statement	Line number	Suspiciousness
1º	type = EQUILATERAL;	3	0.10001
2º	type = RIGI	5	0.10001
3º	type =	7	0.10001
4º	else i	5	0.10000
5º	double	9	0.10000
6º	return	10	0.10000
7º	else if (( ) + (c*c)))	4	0.09999
8º	int type = SCALENE;	1	0.09998
9º	if ((a == b) && (b == c))	2	0.09998
10º	return type; }	8	0.09998



"The **variety** of test cases is the major factor to have **uncertainty in the ranking**"

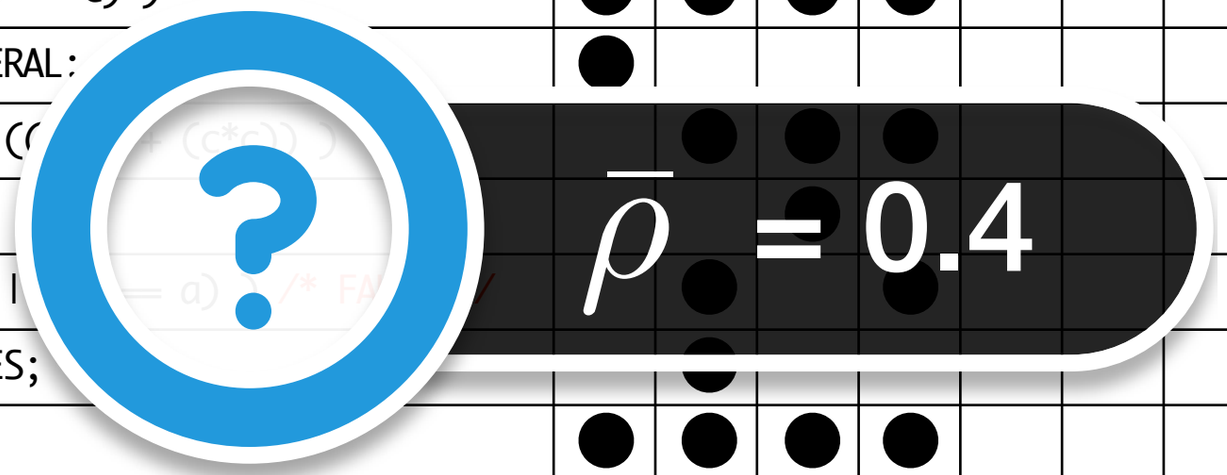
Density of a Test Suite

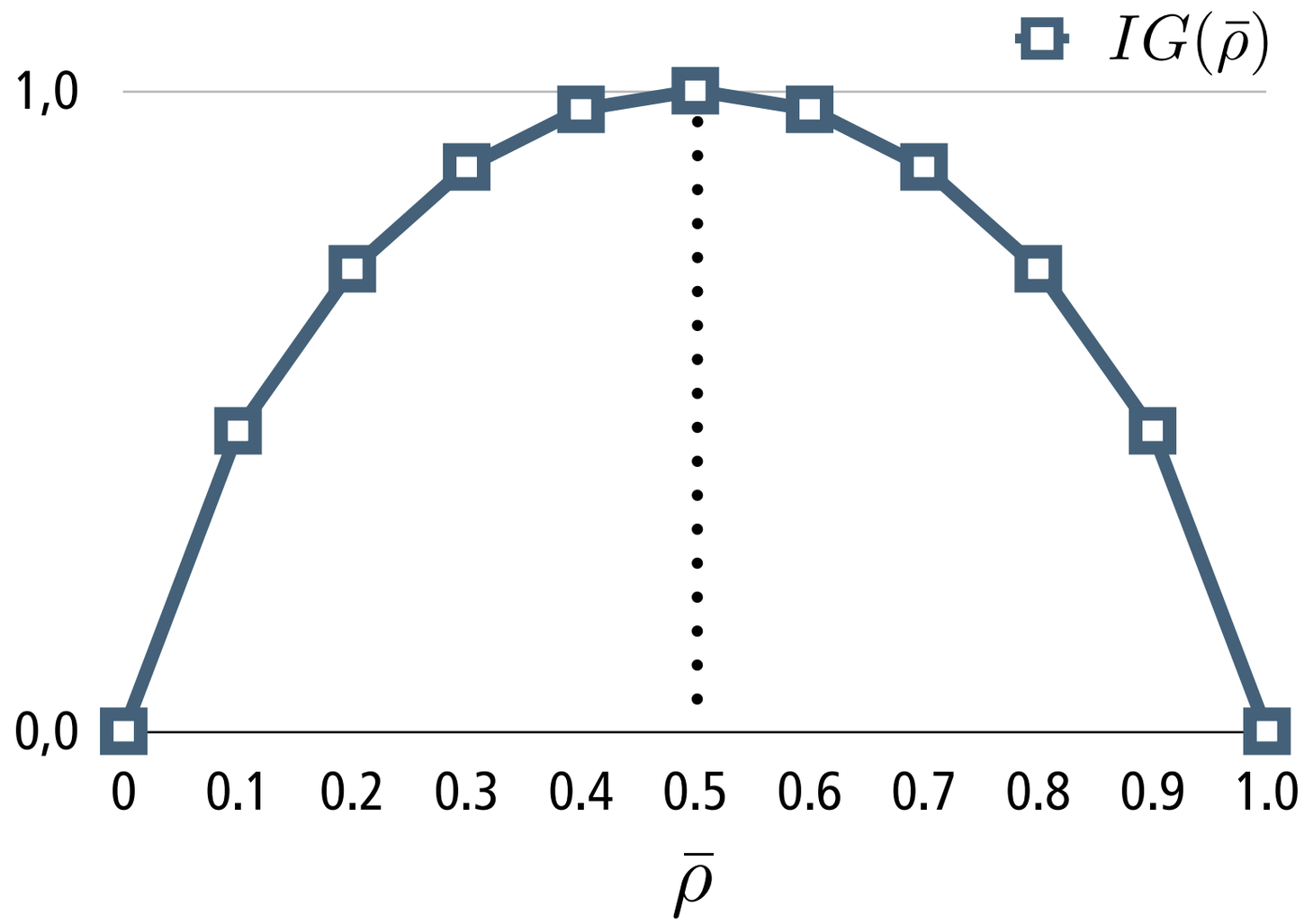
class Triangle {… static int type(int a, int b, int c) {	t_1	t_2	t_3	t_4	t_5	t_6	Suspiciousness
int type = SCALENE;	●	●	●	●			0.09998
if ((a == b) && (b == c))	●	●	●	●			0.09998
type = EQUILATERAL;	●						0.10001
else if ((a*a) == ((b*b) + (c*c)))		●	●	●			0.09999
type = RIGHT;			●				0.10001
else if ((a == b) || (b == a)) /* FAULT */		●		●			**0.10000**
type = ISOSCELES;		●					0.10001
return type; }	●	●	●	●			0.09998
static double area(int a, int b, int c) {							
double s = (a+b+c)/2.0;					●	●	0.10000
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }					●	●	0.10000

A. Gonzalez-Sanchez, R. Abreu, H.-G. Gross, and A. J. van Gemund, "Prioritizing tests for fault localization through ambiguity group reduction", ASE '11

Density of a Test Suite

class Triangle {… static int type(int a, int b, int c) {	t_1	t_2	t_3	t_4	t_5	t_6	Suspiciousness
int type = SCALENE;	●	●	●	●			0.09998
if ((a == b) && (b == c))	●	●	●	●			0.09998
type = EQUILATERAL;	●						0.10001
else if ((a*a) == ((c*c)))		●	●	●			0.09999
type = RIGHT;			●				0.10001
else if ((a == b) || (a == c)) /* FAULT */		●		●			**0.10000**
type = ISOSCELES;		●					0.10001
return type; }	●	●	●	●			0.09998
static double area(int a, int b, int c) {							
double s = (a+b+c)/2.0;					●	●	0.10000
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }					●	●	0.10000



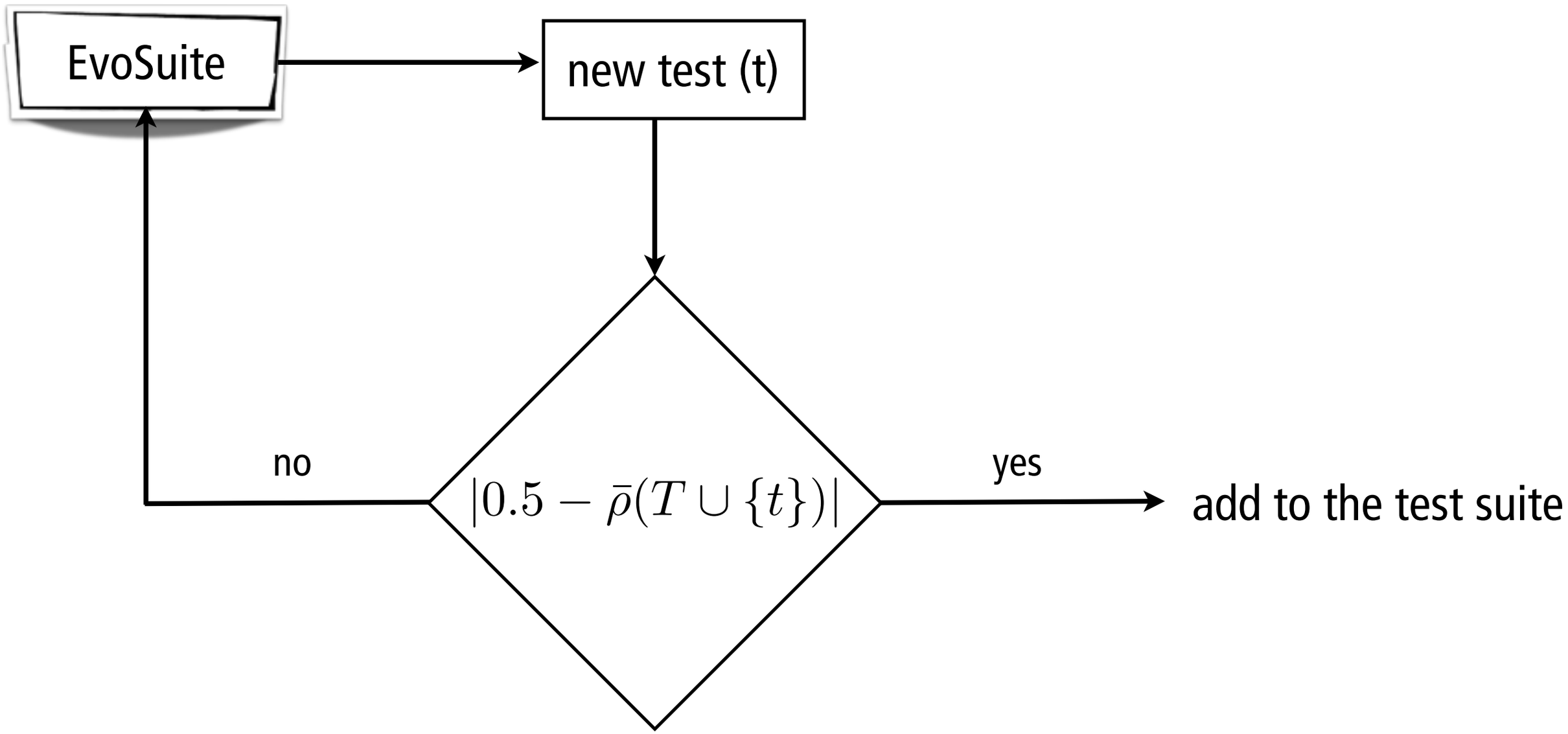


$$IG(\bar{\rho}) = -\bar{\rho} \cdot \log_2(\bar{\rho}) - (1 - \bar{\rho}) \cdot \log_2(1 - \bar{\rho})$$

R. A. Johnson, "An information theory approach to diagnosis", IRE Transactions on Reliability and Quality Control, no. 1, pp. 35–35, 1960

“A fitness function based on **entropy** to guide search-based **test generation** and to optimize the quality of ranking reports”

EntBug


EvoSuite
new test (t)
$|0.5 - \bar{\rho}(T \cup \{t\})|$
no
yes
add to the test suite

	T	
class Triangle {… static int type(int a, int b, int c) {	º	Suspiciousness
int type = SCALENE;	8	0.09998
if ((a == b) && (b == c))	9	0.09998
type = EQUILATERAL;	1	0.10001
else if ((a*a) == ((b*b) + (c*c)))	7	0.09999
type = RIGHT;	2	0.10001
else if ((a == b) || (b == a)) /* FAULT */	**4**	**0.10000**
type = ISOSCELES;	3	0.10001
return type; }	10	0.09998
static double area(int a, int b, int c) {		
double s = (a+b+c)/2.0;	5	0.10000
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }	6	0.10000
Test case outcome (pass = ✓, fail = ✗)		

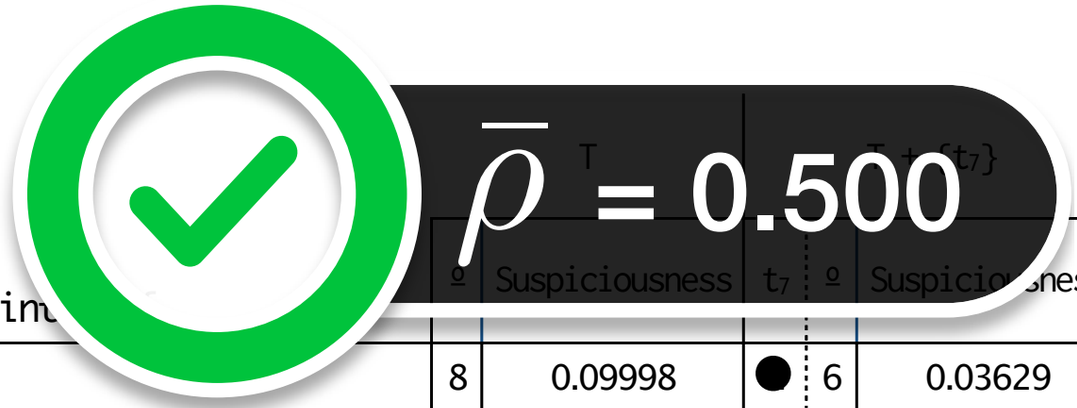
$\bar{\rho}$	0,400
$\mathcal{H}$	3,322
C_d	4,000

	T		T + {t_7}		
class Triangle {… static int type(int a, int b, int c) {	º	Suspiciousness	t_7	º	Suspiciousness
int type = SCALENE;	8	0.09998	●	6	0.03629
if ((a == b) && (b == c))	9	0.09998	●	7	0.03629
type = EQUILATERAL;	1	0.10001			
else if ((a*a) == ((b*b) + (c*c)))	7	0.09999	●	5	0.08466
type = RIGHT;	2	0.10001	●	1	0.29033
else if ((a == b) || (b == a)) /* FAULT */	**4**	**0.10000**	●	**2**	**0.17204**
type = ISOSCELES;	3	0.10001			
return type; }	10	0.09998	●	8	0.03629
static double area(int a, int b, int c) {					
double s = (a+b+c)/2.0;	5	0.10000	●	3	0.17204
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }	6	0.10000	●	4	0.17204
Test case outcome (pass = ✓, fail = ✗)			✗		

$\bar{\rho}$	0,400	0,457
$\mathcal{H}$	3,322	2,651
C_d	4,000	2,000

	T		T + {t_7}			T + {t_7, t_8}		
class Triangle {… static int type(int a, int b, int c) {	º	Suspiciousness	t_7	º	Suspiciousness	t_8	º	Suspiciousness
int type = SCALENE;	8	0.09998	●	6	0.03629	●	6	0.02354
if ((a == b) && (b == c))	9	0.09998	●	7	0.03629	●	7	0.02354
type = EQUILATERAL;	1	0.10001				●		
else if ((a*a) == ((b*b) + (c*c)))	7	0.09999	●	5	0.08466		3	0.10983
type = RIGHT;	2	0.10001	●	1	0.29033		1	0.37666
else if ((a == b) || (b == a)) /* FAULT */	**4**	**0.10000**	●	**2**	**0.17204**		**2**	**0.22320**
type = ISOSCELES;	3	0.10001						
return type; }	10	0.09998	●	8	0.03629	●	8	0.02354
static double area(int a, int b, int c) {								
double s = (a+b+c)/2.0;	5	0.10000	●	3	0.17204	●	4	0.10983
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }	6	0.10000	●	4	0.17204	●	5	0.10983
Test case outcome (pass = ✓, fail = ✗)			✗			✓		
$\bar{\rho}$	0,400		0,457			0,475		
$\mathcal{H}$	3,322		2,651			2,445		
C_d	4,000		2,000			1,000		

	T		T + {t_7}			T + {t_7, t_8}			T + {t_7, t_8, t_9}		
class Triangle {… static int type(int a, int b, int c) {	º	Suspiciousness	t_7	º	Suspiciousness	t_8	º	Suspiciousness	t_9	º	Suspiciousness
int type = SCALENE;	8	0.09998	●	6	0.03629	●	6	0.02354	●	5	0.04347
if ((a == b) && (b == c))	9	0.09998	●	7	0.03629	●	7	0.02354	●	6	0.04347
type = EQUILATERAL;	1	0.10001				●					
else if ((a*a) == ((b*b) + (c*c)))	7	0.09999	●	5	0.08466		3	0.10983	●	2	0.17391
type = RIGHT;	2	0.10001	●	1	0.29033		1	0.37666			
else if ((a == b) || (b == a)) /* FAULT */	**4**	**0.10000**	●	**2**	**0.17204**		**2**	**0.22320**	●	**1**	**0.34782**
type = ISOSCELES;	3	0.10001									
return type; }	10	0.09998	●	8	0.03629	●	8	0.02354	●	7	0.04347
static double area(int a, int b, int c) {											
double s = (a+b+c)/2.0;	5	0.10000	●	3	0.17204	●	4	0.10983	●	3	0.17391
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }	6	0.10000	●	4	0.17204	●	5	0.10983	●	4	0.17391
Test case outcome (pass = ✓, fail = ✗)			✗			✓			✗		
$\bar{\rho}$	0,400		0,457			0,475			**0,500**		
$\mathcal{H}$	3,322		2,651			2,445			**2,437**		
C_d	4,000		2,000			1,000			**0,000**		



	T		T + {t_7}		T + {t_7, t_8}		T + {t_7, t_8, t_9}	
class Triangle {… static int type(int a, int b, int	º	Suspiciousness	t_7	º	Suspiciousness	t_8	º	Suspiciousness	t_9	º	Suspiciousness
int type = SCALENE;	8	0.09998	●	6	0.03629	●	6	0.02354	●	5	0.04347
if ((a == b) && (b == c))	9	0.09998	●	7	0.03629	●	7	0.02354	●	6	0.04347
type = EQUILATERAL;	1	0.10001				●					
else if ((a*a) == ((b*b) + (c*c)))	7	0.09999	●	5	0.08466		3	0.10983	●	2	0.17391
type = RIGHT;	2	0.10001	●	1	0.29033		1	0.37666			
else if ((a == b) || (b == a)) /* FAULT */	**4**	**0.10000**	●	**2**	**0.17204**		**2**	**0.22320**	●	**1**	**0.34782**
type = ISOSCELES;	3	0.10001									
return type; }	10	0.09998	●	8	0.03629	●	8	0.02354	●	7	0.04347
static double area(int a, int b, int c) {											
double s = (a+b+c)/2.0;	5	0.10000	●	3	0.17204	●	4	0.10983	●	3	0.17391
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }	6	0.10000	●	4	0.17204	●	5	0.10983	●	4	0.17391
Test case outcome (pass = ✓, fail = ✗)			✗			✓			✗		

$\bar{\rho}$	0,400	0,457	0,475	**0,500**
$\mathcal{H}$	3,322	2,651	2,445	**2,437**
C_d	4,000	2,000	1,000	**0,000**

$\bar{\rho} = 0.500$

$\mathcal{H} \downarrow$ -27%

	T			T + {t_7}			T + {t_7, t_8}			T + {t_7, t_8, t_9}		
class Triangle {… static int type(int a, int b, int	º	Suspiciousness	t_7	º	Suspiciousness	t_8	º	Suspiciousness	t_9	º	Suspiciousness
int type = SCALENE;	8	0.09998	●	6	0.03629	●	6	0.02354	●	5	0.04347
if ((a == b) && (b == c))	9	0.09998	●	7	0.03629	●	7	0.02354	●	6	0.04347
type = EQUILATERAL;	1	0.10001				●					
else if ((a*a) == ((b*b) + (c*c)))	7	0.09999	●	5	0.08466		3	0.10983	●	2	0.17391
type = RIGHT;	2	0.10001	●	1	0.29033		1	0.37666			
else if ((a == b) || (b == */	4	0.10000	●	2	0.17204		2	0.22320	●	1	0.34782
type = ISOSCELES;	3	0.10001									
return type; }	10	0.09998	●	8	0.03629	●	8	0.02354	●	7	0.04347
static double area(int a, int											
double s = (a+b+c)/2.0;	5	0.10000	●	3	0.17204	●	4	0.10983	●	3	0.17391
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }	6	0.10000	●	4	0.17204	●	5	0.10983	●	4	0.17391
Test case outcome (pass = ✓, fail = ✗)			✗			✓			✗		
$\bar{\rho}$	0,400		0,457		0,475			0,500			
$\mathcal{H}$	3,322		2,651		2,445			2,437			
C_d	4,000		2,000		1,000			0,000			

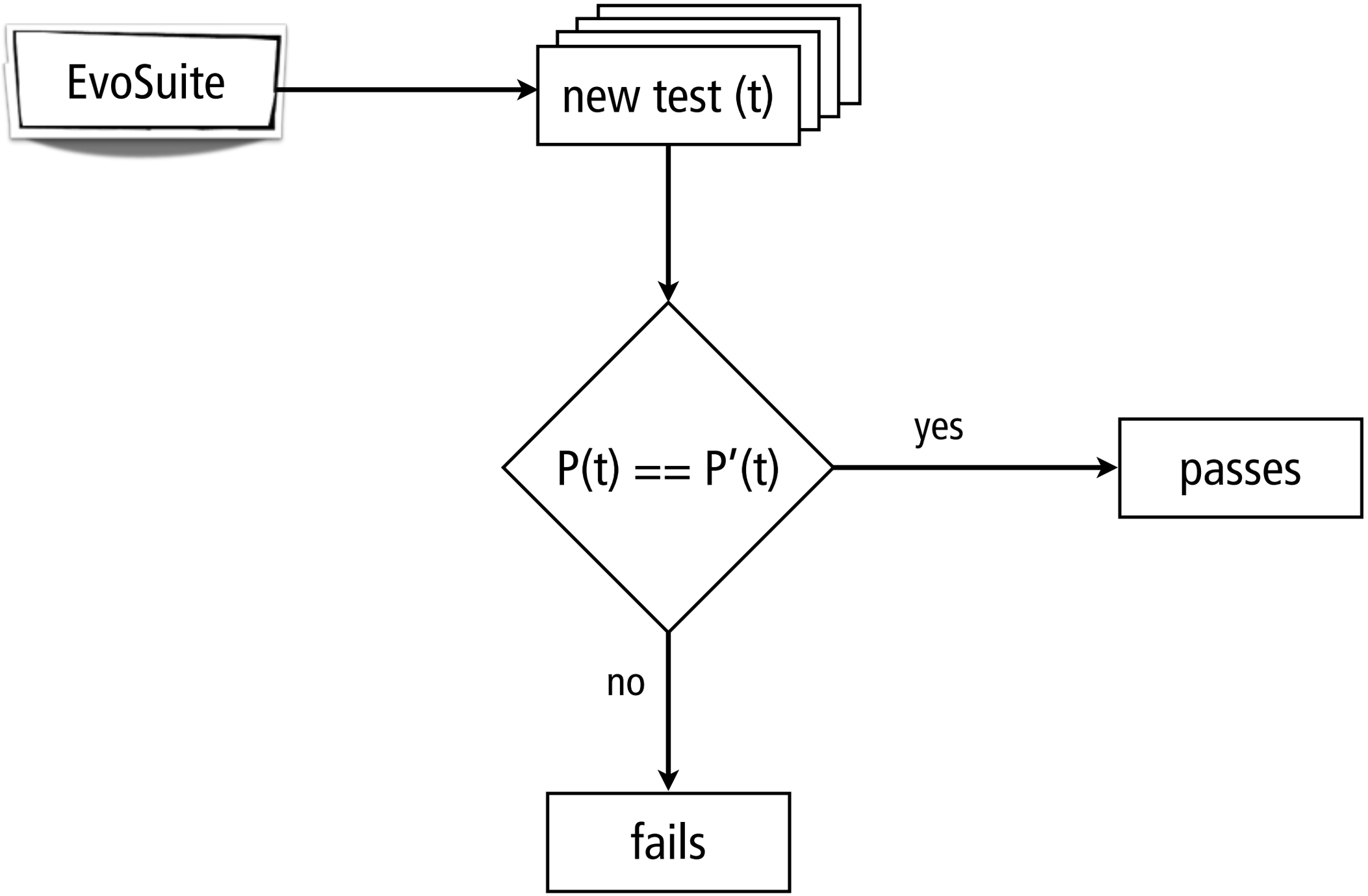
$\bar{\rho} = 0.500$

$\mathcal{H}\downarrow$ -27%

$C_d = 0.0$

class Triangle {…	T		T + {t_7}			T + {t_7, t_8}			T + {t_7, t_8, t_9}		
static int type(int a, int b, int	º	Suspiciousness	t_7	º	Suspiciousness	t_8	º	Suspiciousness	t_9	º	Suspiciousness
int type = SCALENE;	8	0.09998	●	6	0.03629	●	6	0.02354	●	5	0.04347
if ((a == b) && (b == c))	9	0.09998	●	7	0.03629	●	7	0.02354	●	6	0.04347
type = EQUILATERAL;	1	0.10001				●					
else if ((a*a) == ((b*b) + (c*c)))	7	0.09999	●	5	0.08466		3	0.10983	●	2	0.17391
type = RIGHT;	2	0.10001	●	1	0.29033		1	0.37666			
else if ((a == b) || (b == */	4	0.10000	●	2	0.17204		2	0.22320	●	1	0.34782
type = ISOSCELES;	3	0.10001									
return type; }	10	0.09998	●	8	0.03629	●	8	0.02354	●	7	0.04347
static double area(int a, int											
double s = (a+b+c)/2.0;	5	0.10000	●	3	0.17204	●	4	0.10983	●	3	0.17391
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }	6	0.10000	●	4	0.17204	●	5	0.10983	●	4	0.17391
Test case outcome (pass = ✓, fail = ✗)			✗			✓			✗		
$\bar{\rho}$	0,400		0,457			0,475			0,500		
$\mathcal{H}$	3,322		2,651			2,445			2,437		
C_d	4,000		2,000			1,000			0,000		

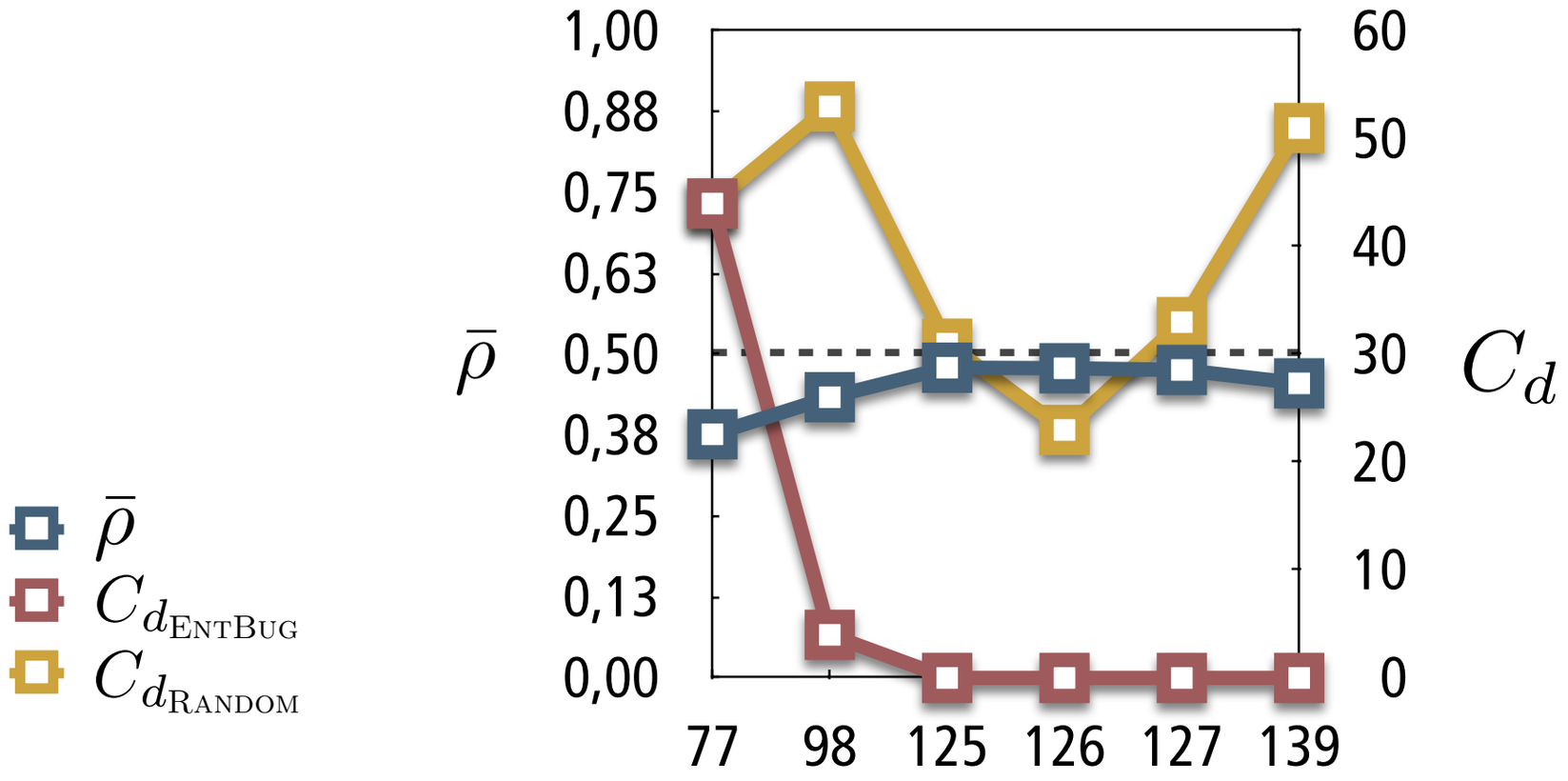
EVALUATION - SETUP

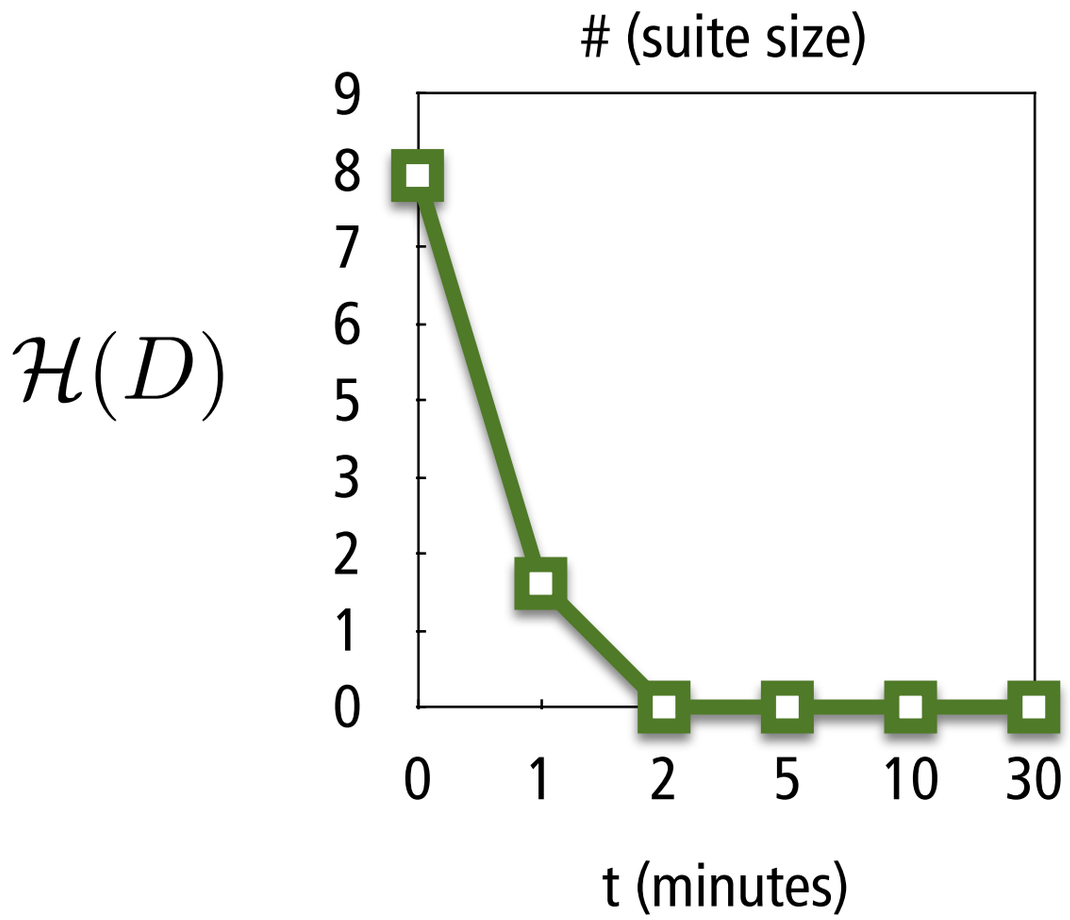


Evaluation - Subjects

Subject	Revision	Classes	LOCs	Original Test Suite / Test Cases Used	BugID
Vending Machine	-	2	54	1 / 1	-
Apache Com. Codec	928 140	24	2 998	303 / 77	99
Apache Com. Compress	768 548	62	7 365	121 / 26	114
Apache Com. Math	1 244 107	537	61 921	3,541 / 197	835
Apache Com. Math	1 416 643	588	69 520	4,318 / 26	938
Apache Com. Math	1 416 643	588	69 520	4,318 / 12	939
Joda Time	1 070	188	23 964	2,921 / 169	-

Evaluation - Apache Commons Codec #99

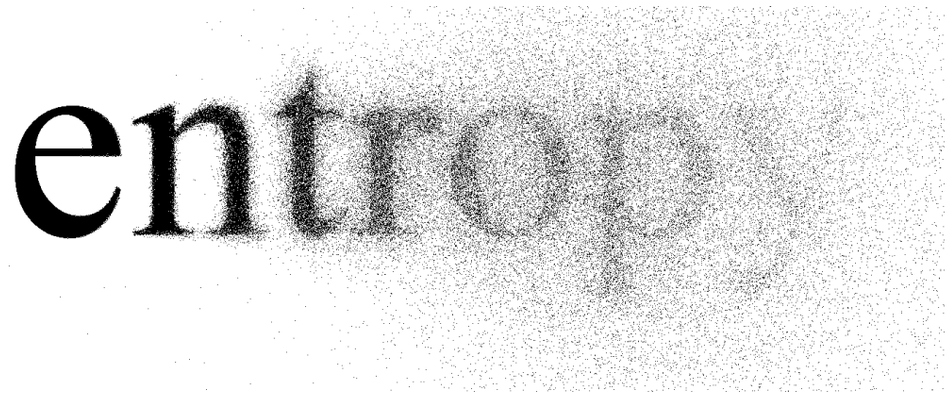




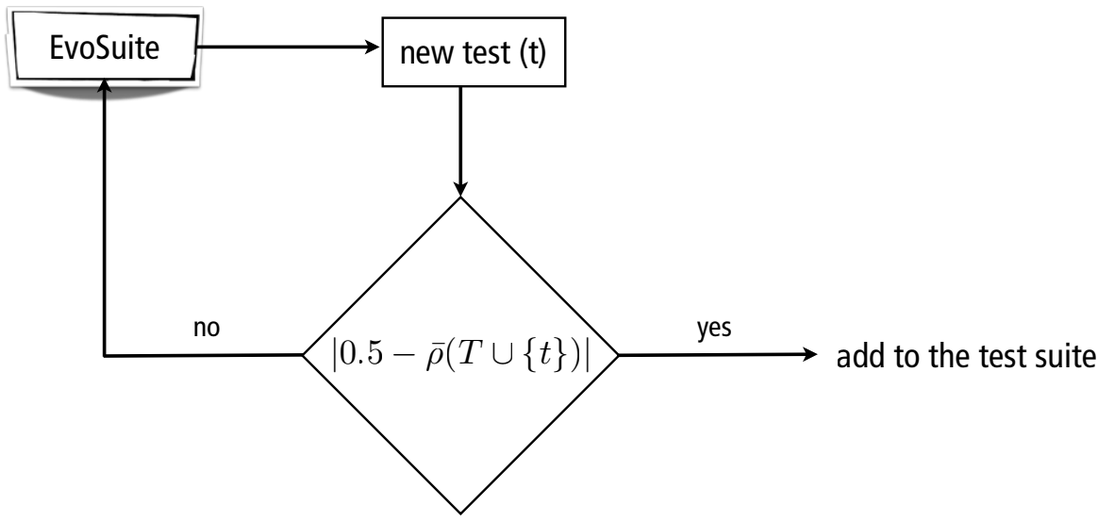
CONTRIBUTIONS

1. A fitness function based on entropy to guide search-based test generation and to optimize the quality of ranking reports;
2. A prototype implementation of the described approach on top of the EVOSUITE test generation tool;
3. An evaluation of the approach using six real faults;
4. Empirical results show that ENTBUG reduces the $\mathcal{H}$ by 49% on average, leading to a 91% average reduction of diagnosis candidates needed to inspect to find the true faulty one.

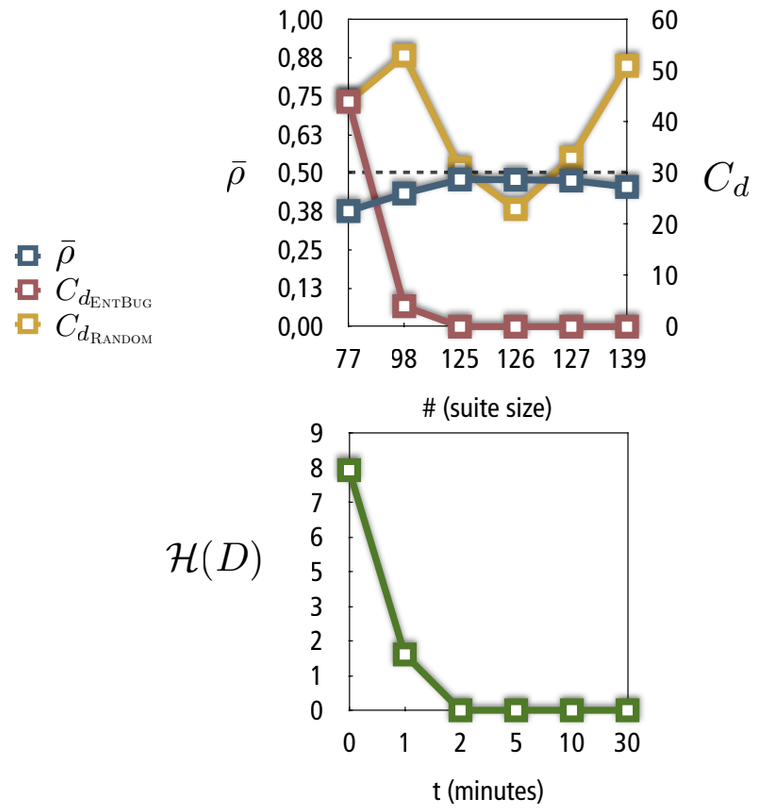
class Triangle {… static int type(int a, int b, int c) {	t_1	t_2	t_3	t_4	t_5	t_6	Suspiciousness
int type = SCALENE;	●	●	●	●			0.09998
if ((a == b) && (b == c))	●	●	●	●			0.09998
type = EQUILATERAL;	●						0.10001
else if ((a*a) == ((b*b) + (c*c)))		●	●	●			0.09999
type = RIGHT;			●				0.10001
else if ((a == b) || (b == a)) /* FAULT */		●		●			**0.10000**
type = ISOSCELES;		●					0.10001
return type; }	●	●	●	●			0.09998
static double area(int a, int b, int c) {							
double s = (a+b+c)/2.0;					●	●	0.10000
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }					●	●	0.10000



EntBug



Evaluation - Apache Commons Codec #99

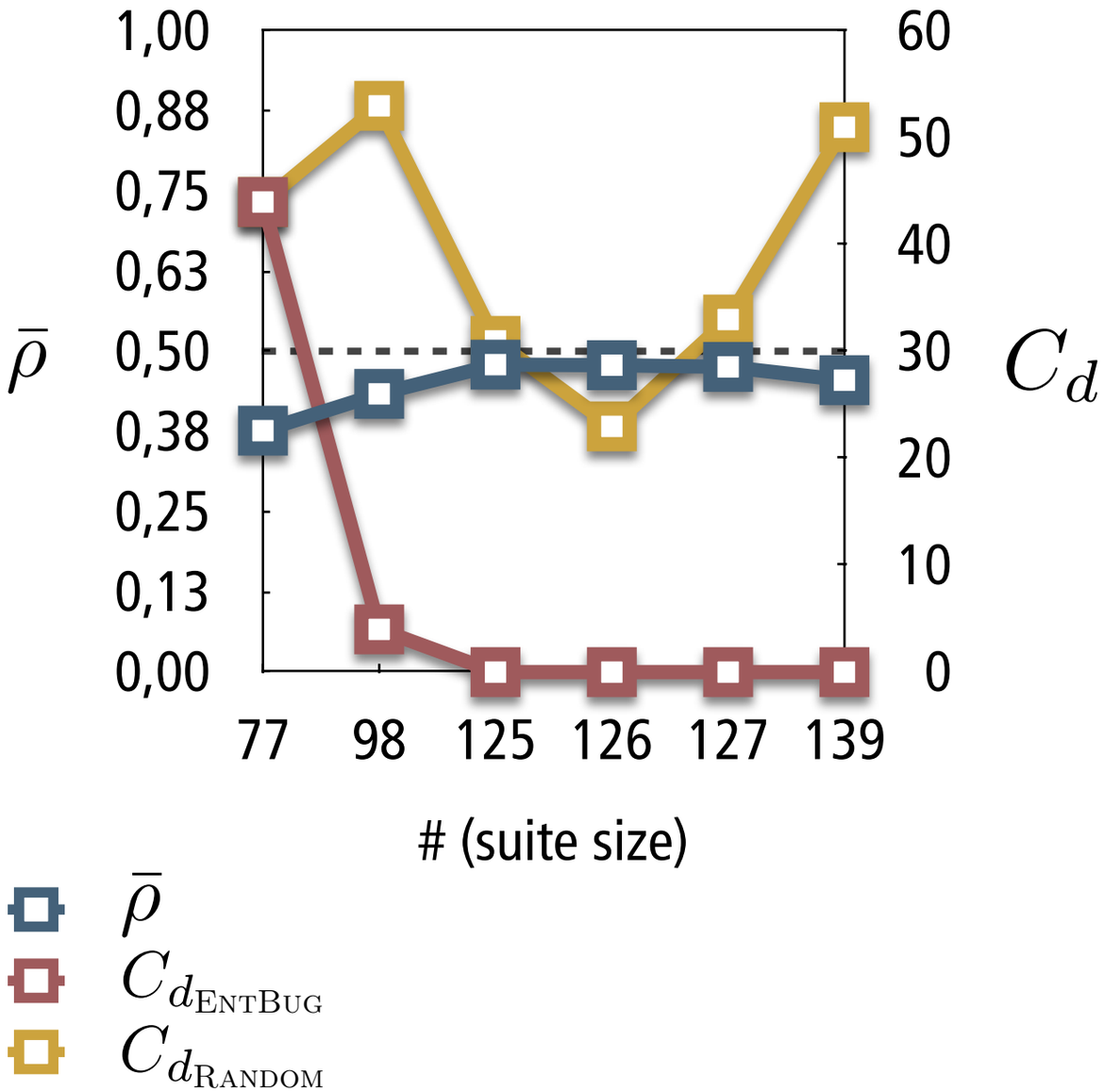


Annex

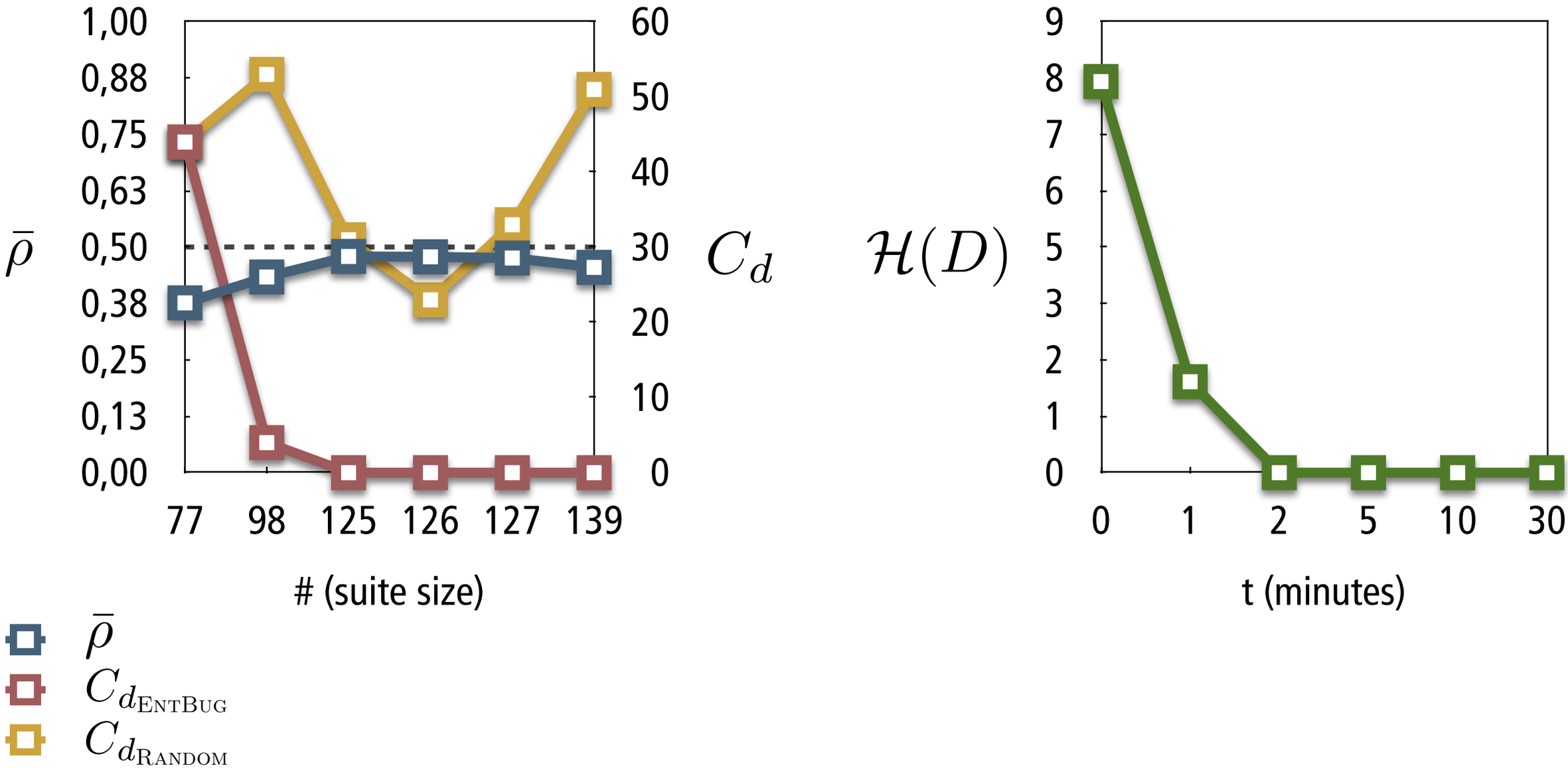
Evaluation - Apache Commons Codec #99

```
// org.apache.commons.codec.binary.Base64
@@ -667,7 +667,7 @@
  public static String encodeBase64String(byte[] binaryData) {
-   return StringUtils.newStringUtf8(encodeBase64(binaryData, true));
+   return StringUtils.newStringUtf8(encodeBase64(binaryData, false));
  } ...
```

Evaluation - Apache Commons Codec #99



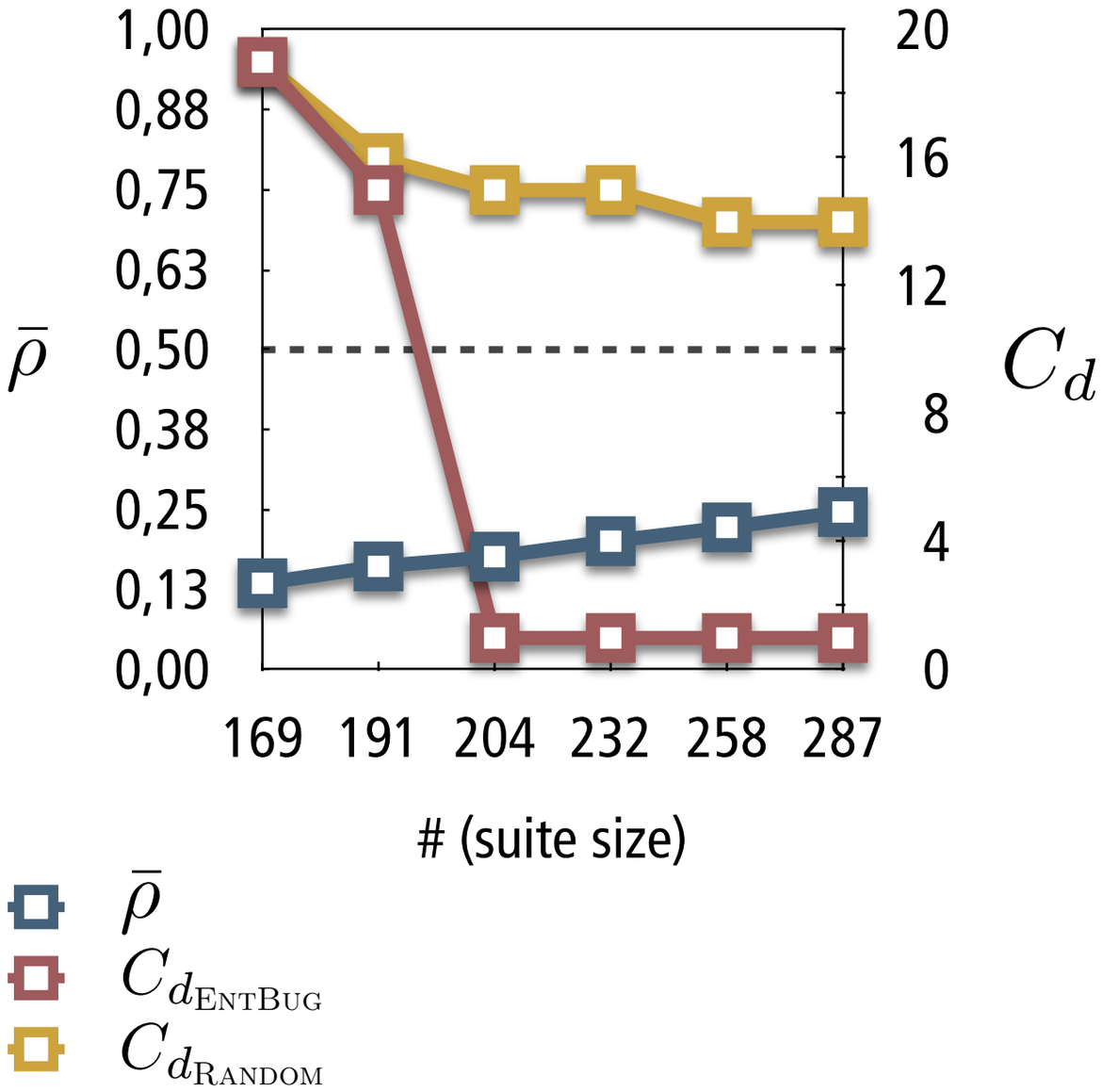
Evaluation - Apache Commons Codec #99



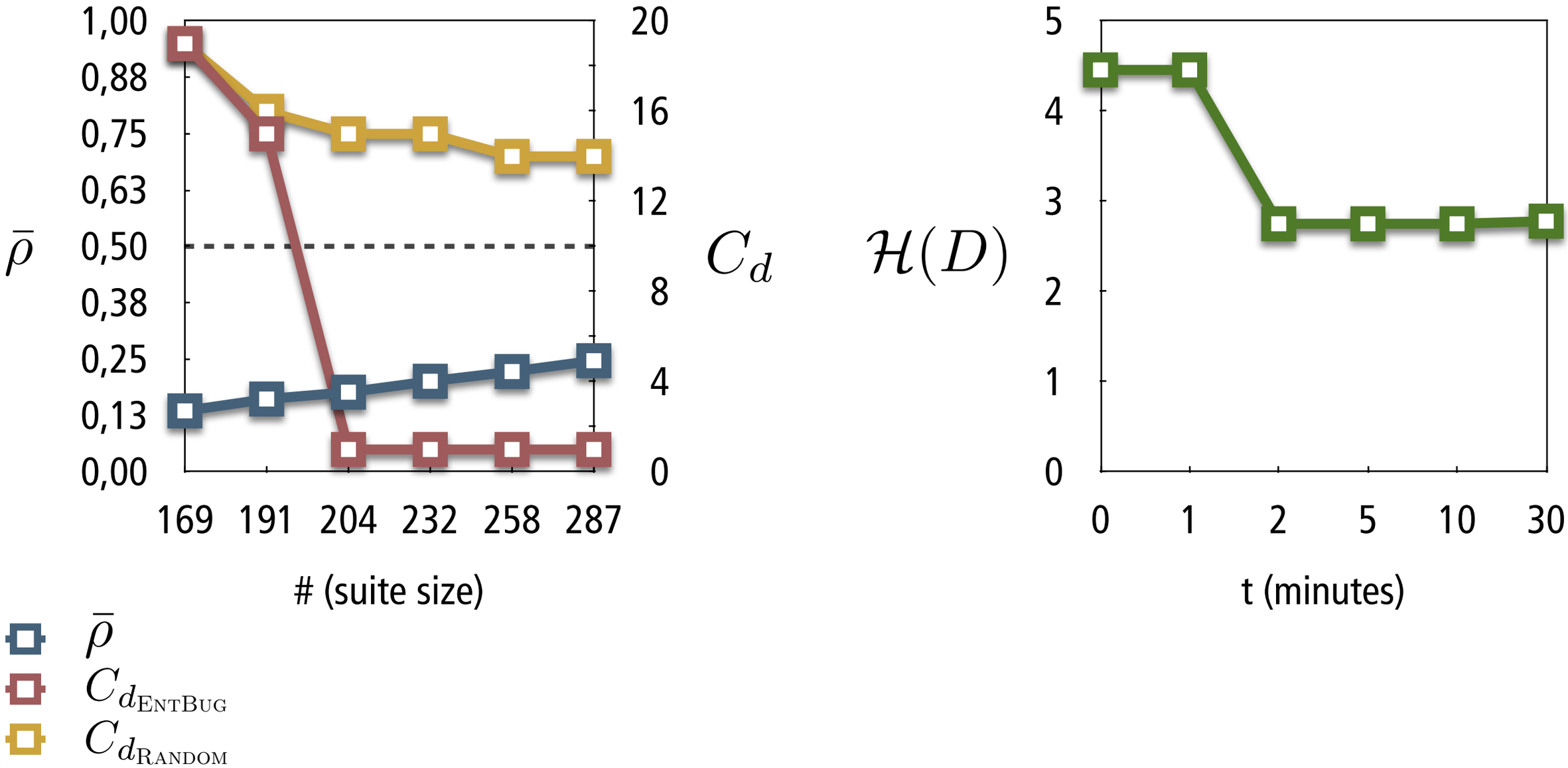
Evaluation - Joda Time

```
// org.joda.time.chrono.BasicDayOfYearDateTimeField
@@ -90,7 +90,7 @@
  protected int getMaximumValueForSet(long instant, int value) {
    int maxLessOne = iChronology.getDaysInYearMax() - 1;
-   return value > maxLessOne ? getMaximumValue(instant) : maxLessOne;
+   return (value > maxLessOne || value < 1) ? getMaximumValue(instant)
      : maxLessOne;
  } ...
```

Evaluation - Joda Time



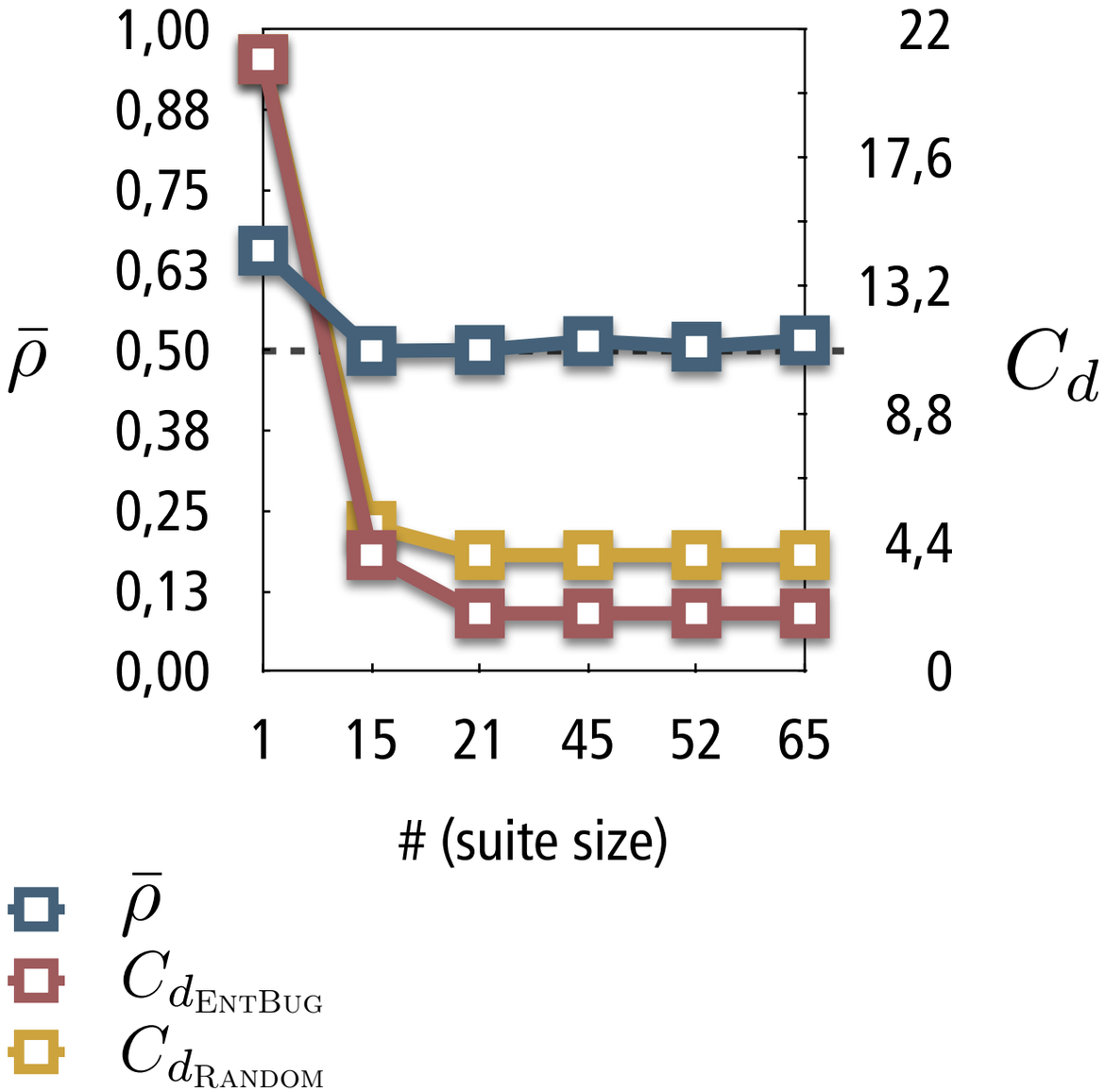
Evaluation - Joda Time



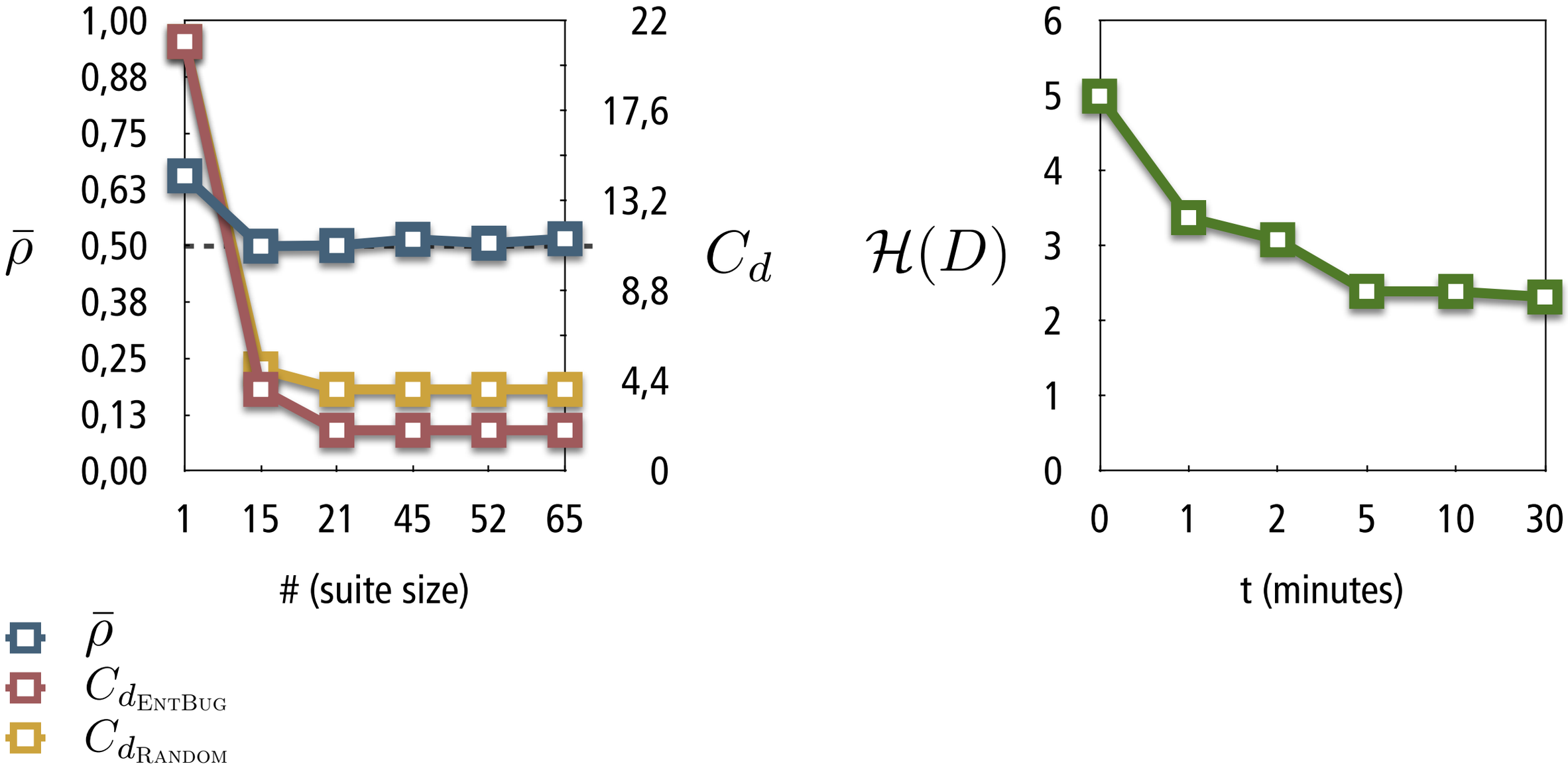
Evaluation - Vending Machine

```
// Class : vendingmachine.VendingMachine
@@ -45,7 +45,7 @@
  public void vend() throws Exception { ...
    this.currValue -= COST;
-   if (this.currValue == 0) {
+   if ((this.currValue - COST) <= 0) {
      this.enabled = false;
    } ...
```

Evaluation - Vending Machine



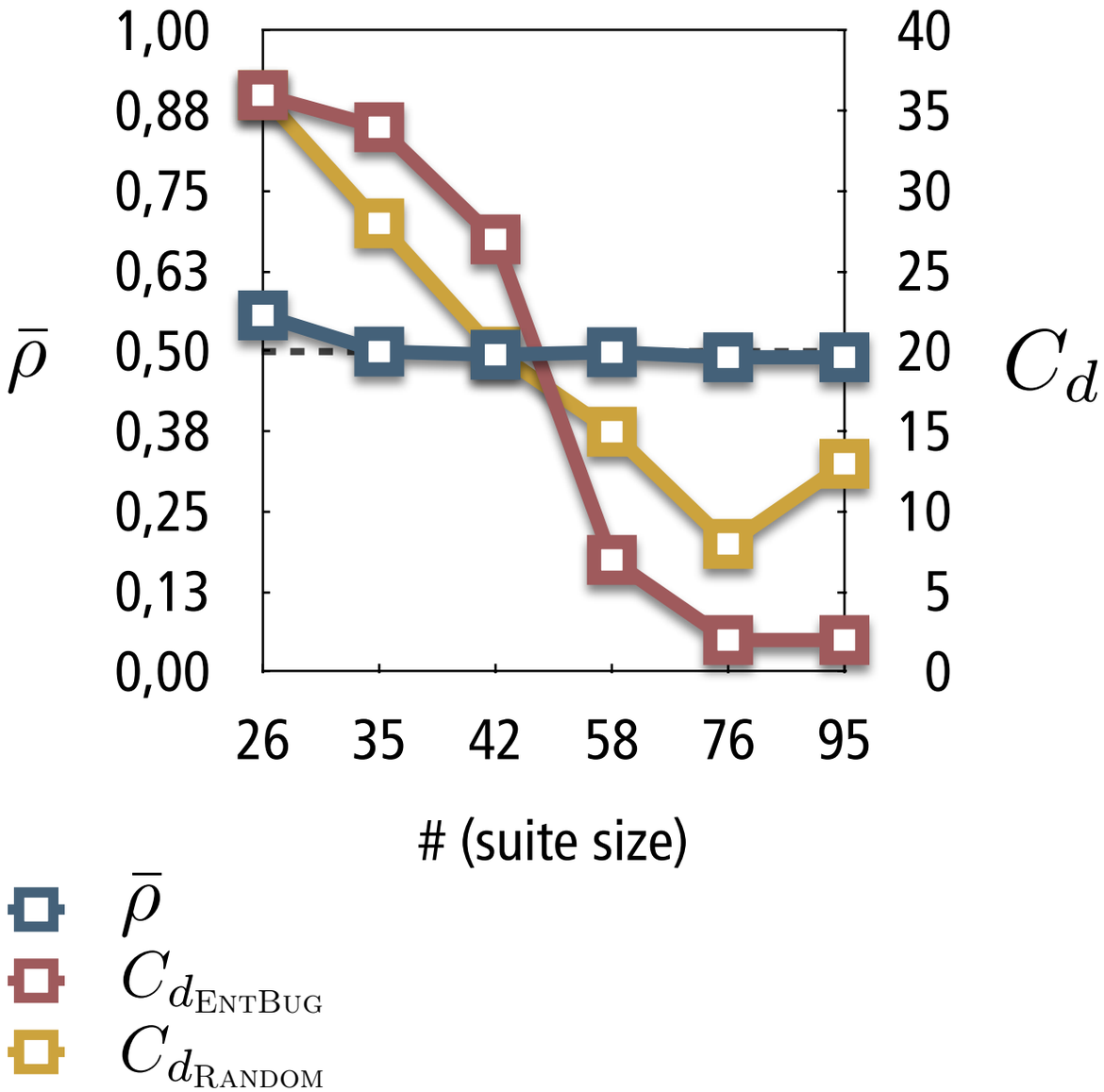
Evaluation - Vending Machine



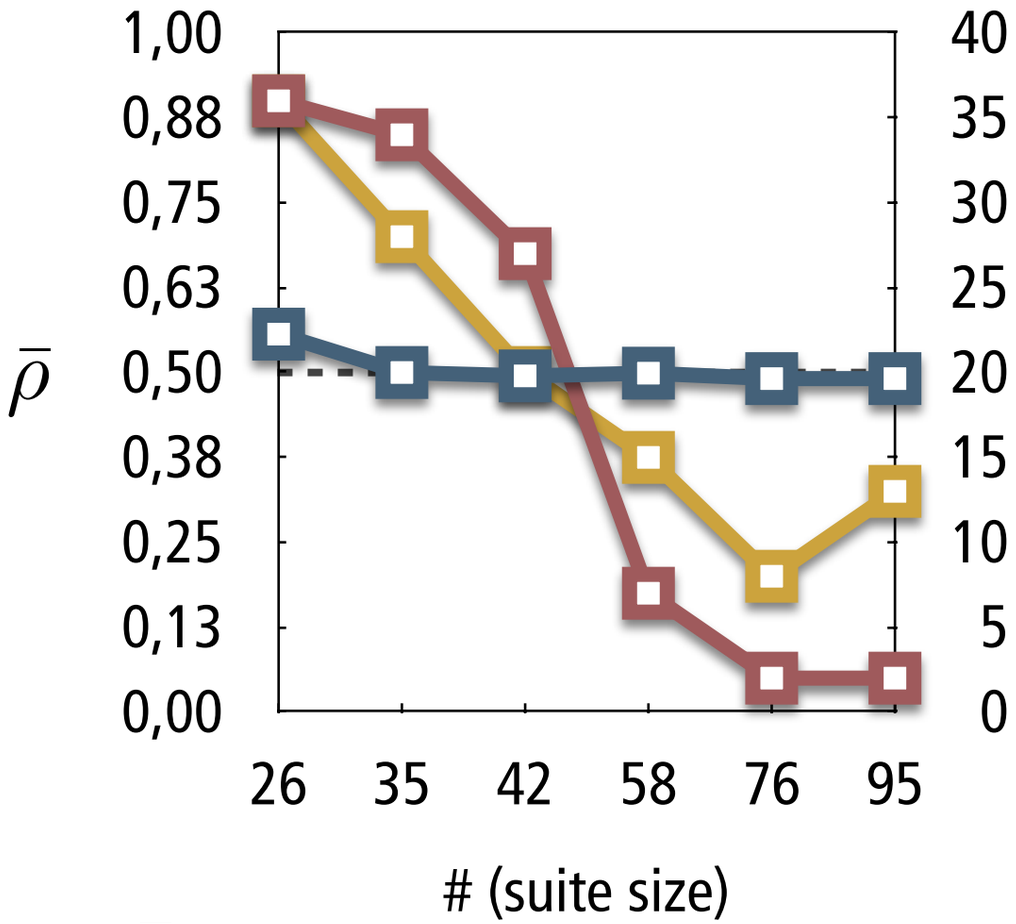
Evaluation - Apache Commons Compress #114

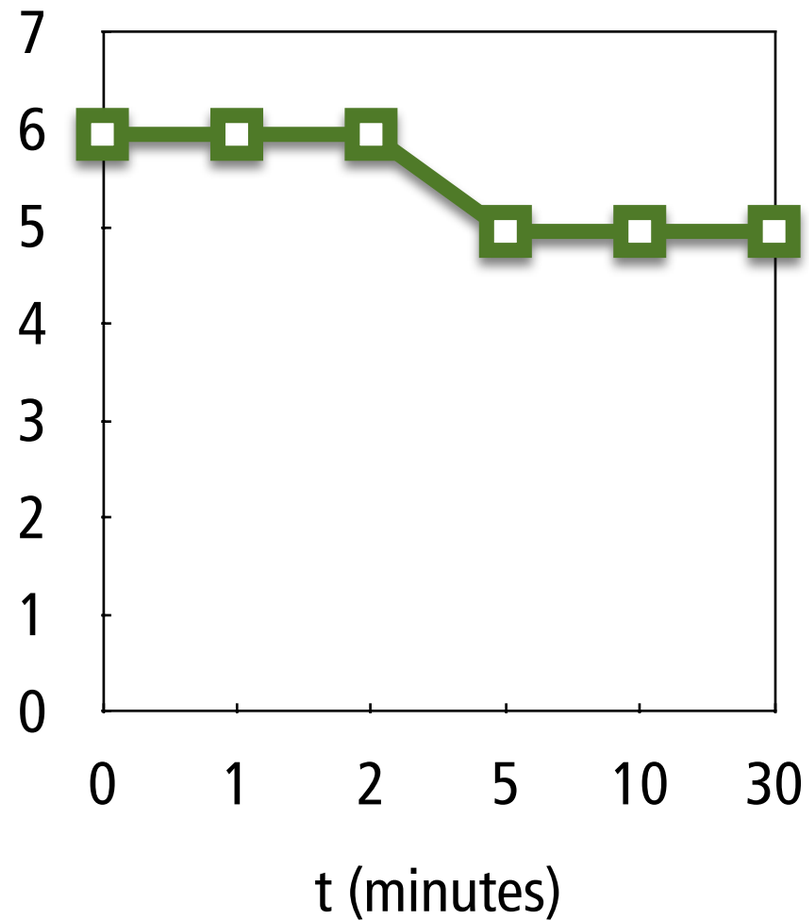
```
// org.apache.commons.compress.archivers.tar.TarUtils
@@ -95,11 +95,11 @@
  for (int i = offset; i < end; ++i) { ...
-   result.append((char) buffer[i]);
+   result.append((char) (b & 0xFF));//Allow for sign extension
  } ...
```

Evaluation - Apache Commons Compress #114



Evaluation - Apache Commons Compress #114

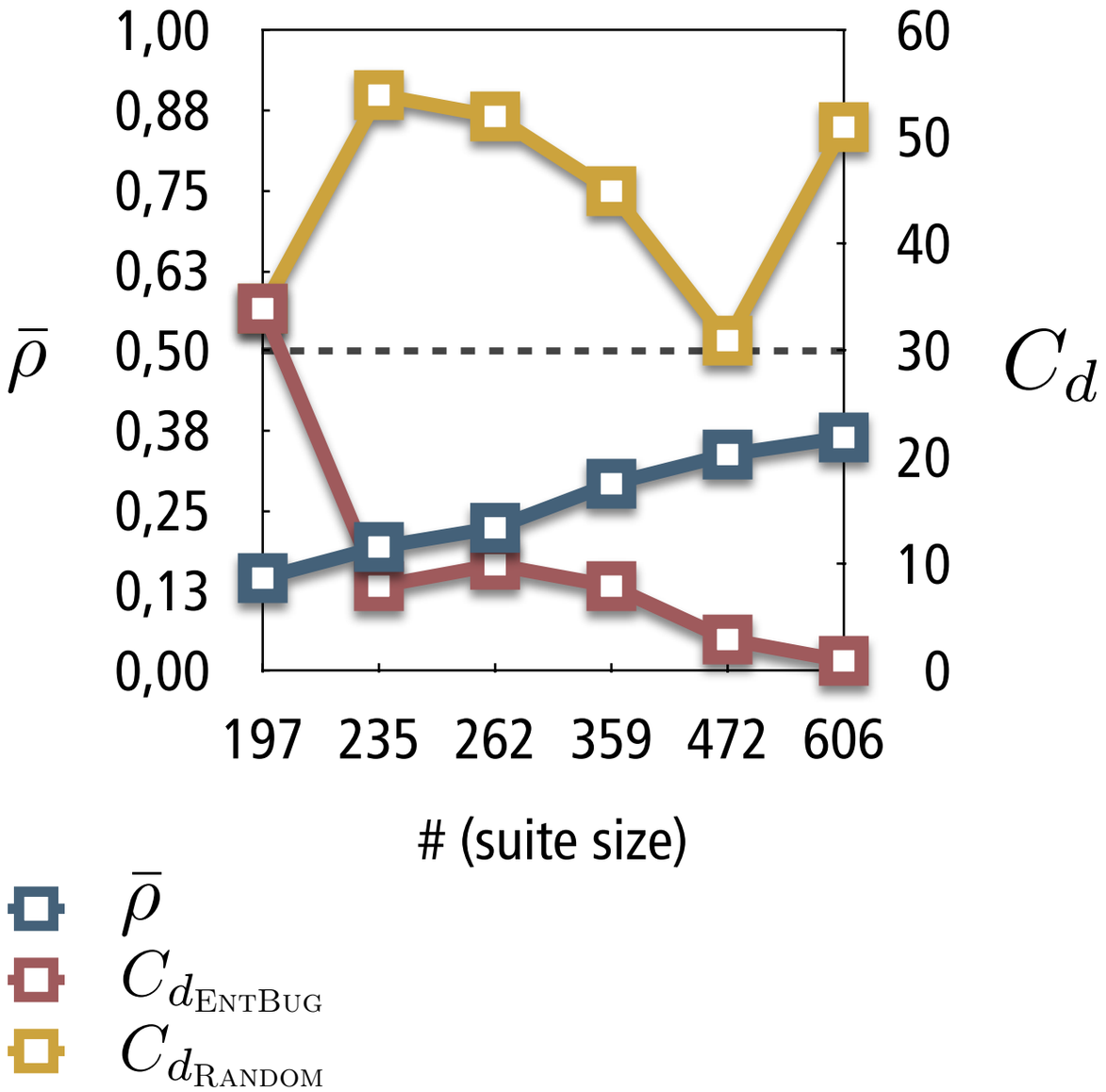




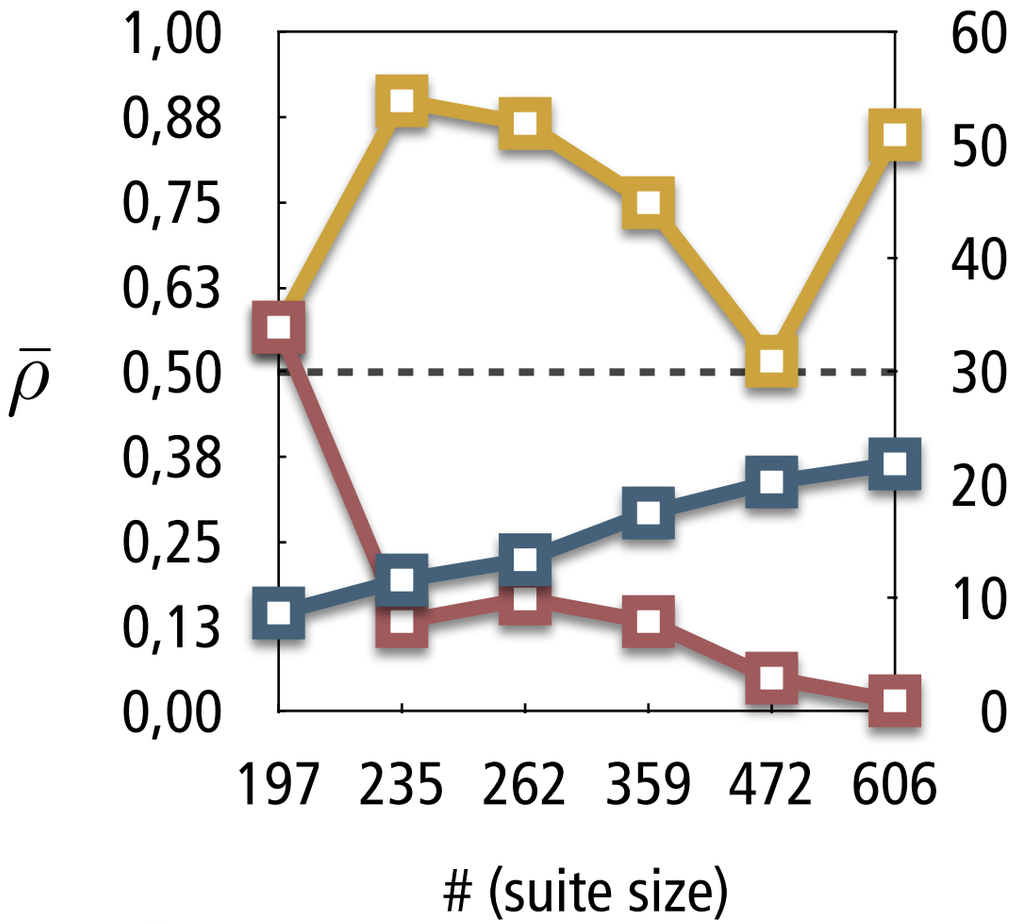
Evaluation - Apache Commons Math #835

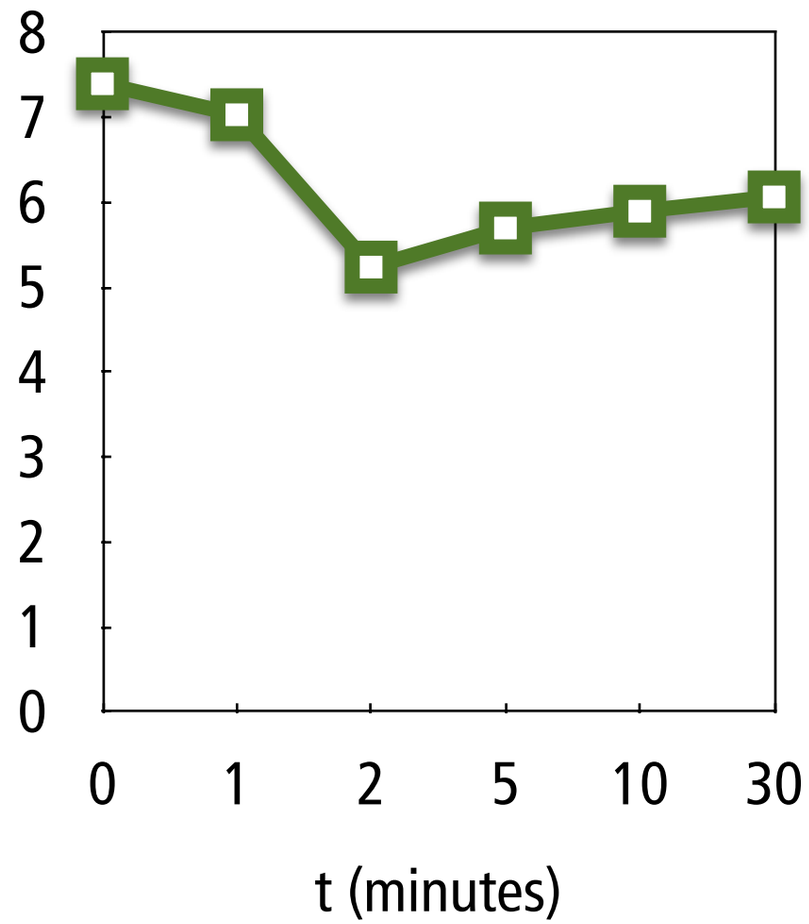
```
// org.apache.commons.math3.fraction.Fraction
@@ -594,7 +594,7 @@
  public double percentageValue() {
-   return multiply(100).doubleValue();
+   return 100 * doubleValue();
  } ...
```

Evaluation - Apache Commons Math #835



Evaluation - Apache Commons Math #835

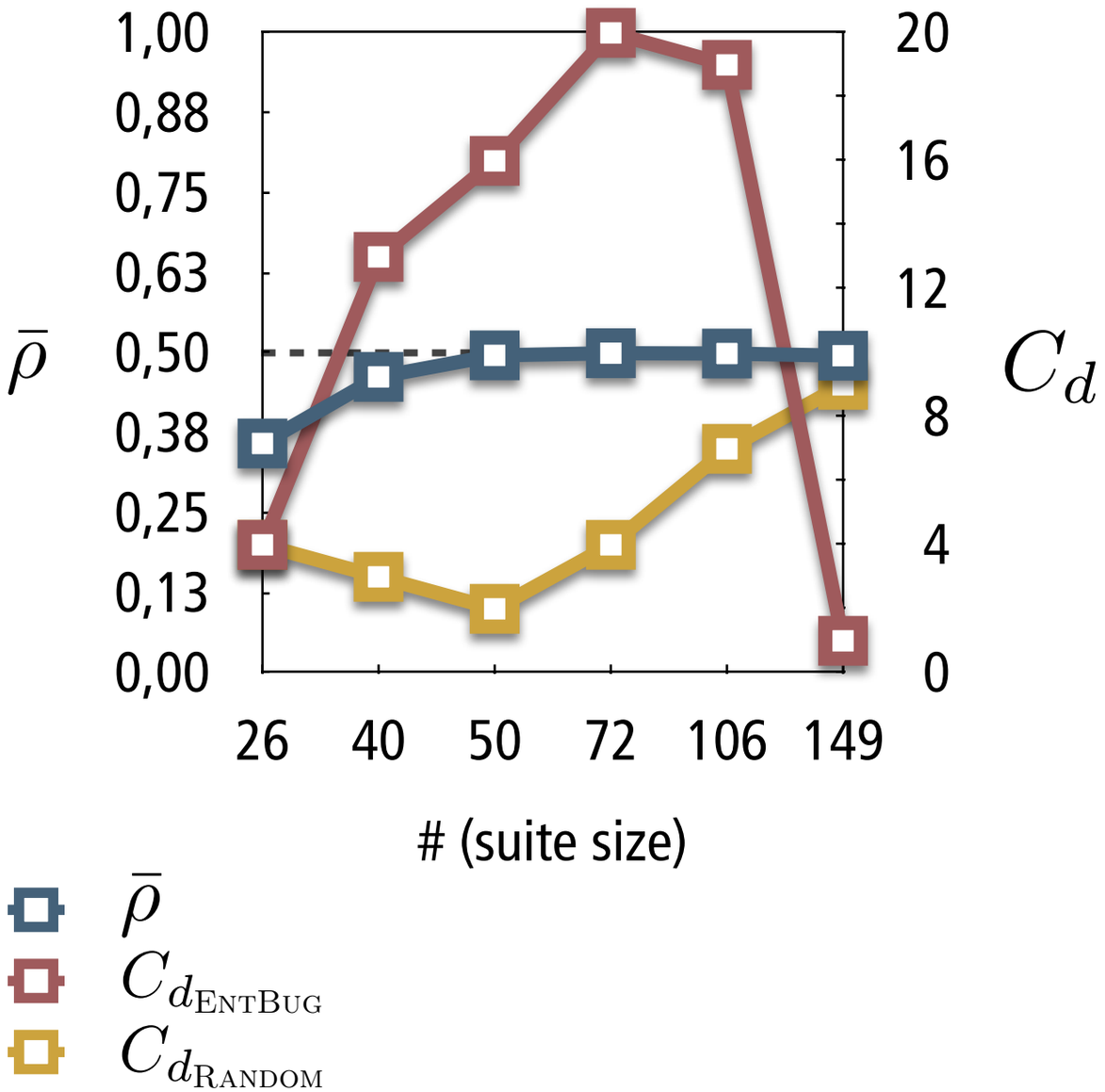




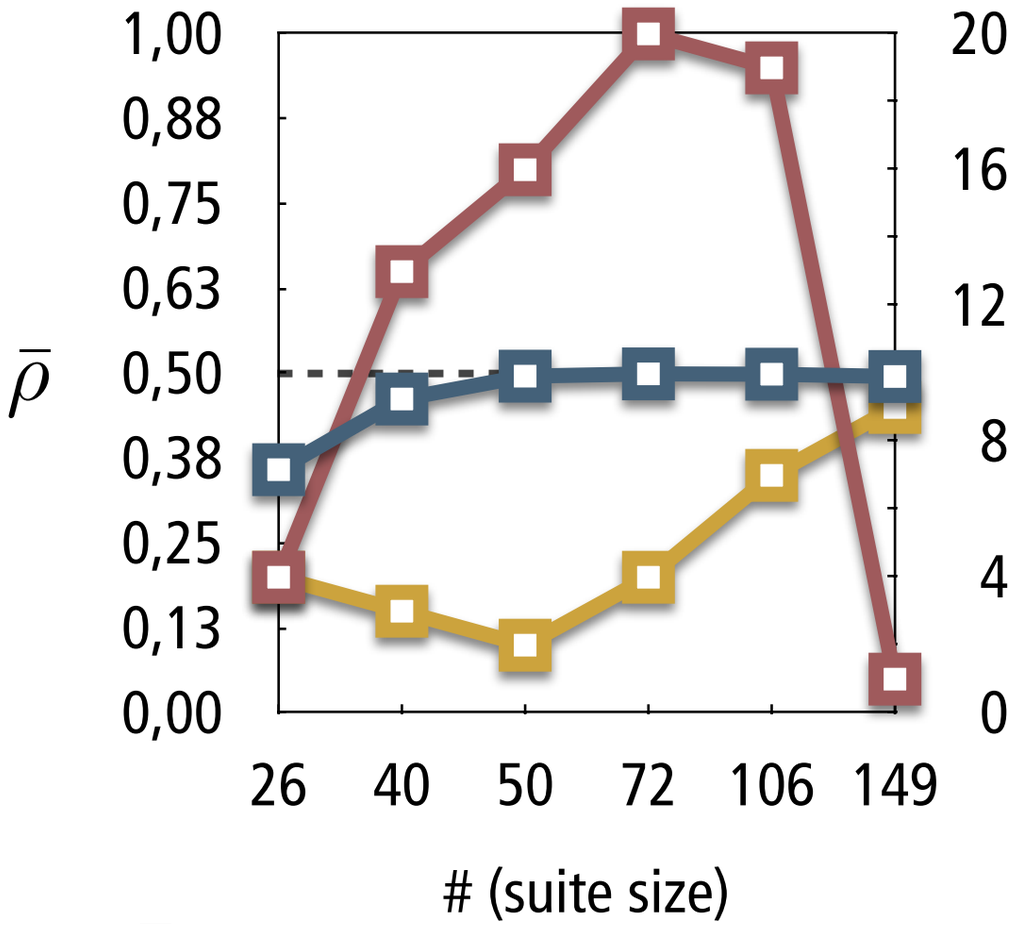
Evaluation - Apache Commons Math #938

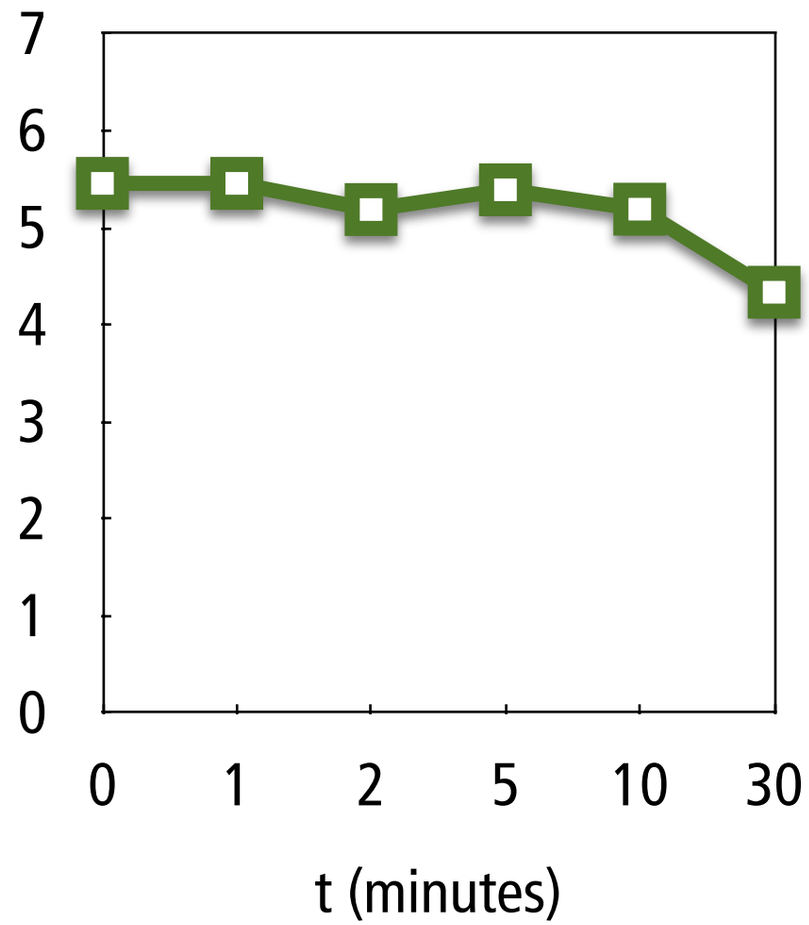
```
// org.apache.commons.math3.geometry.euclidean.threed.Line
@@ -84,7 +84,9 @@
  public Line revert() {
-   return new Line(zero, zero.subtract(direction));
+   final Line reverted = new Line(this);
+   reverted.direction = reverted.direction.negate();
+   return reverted;
  } ...
```

Evaluation - Apache Commons Math #938



Evaluation - Apache Commons Math #938

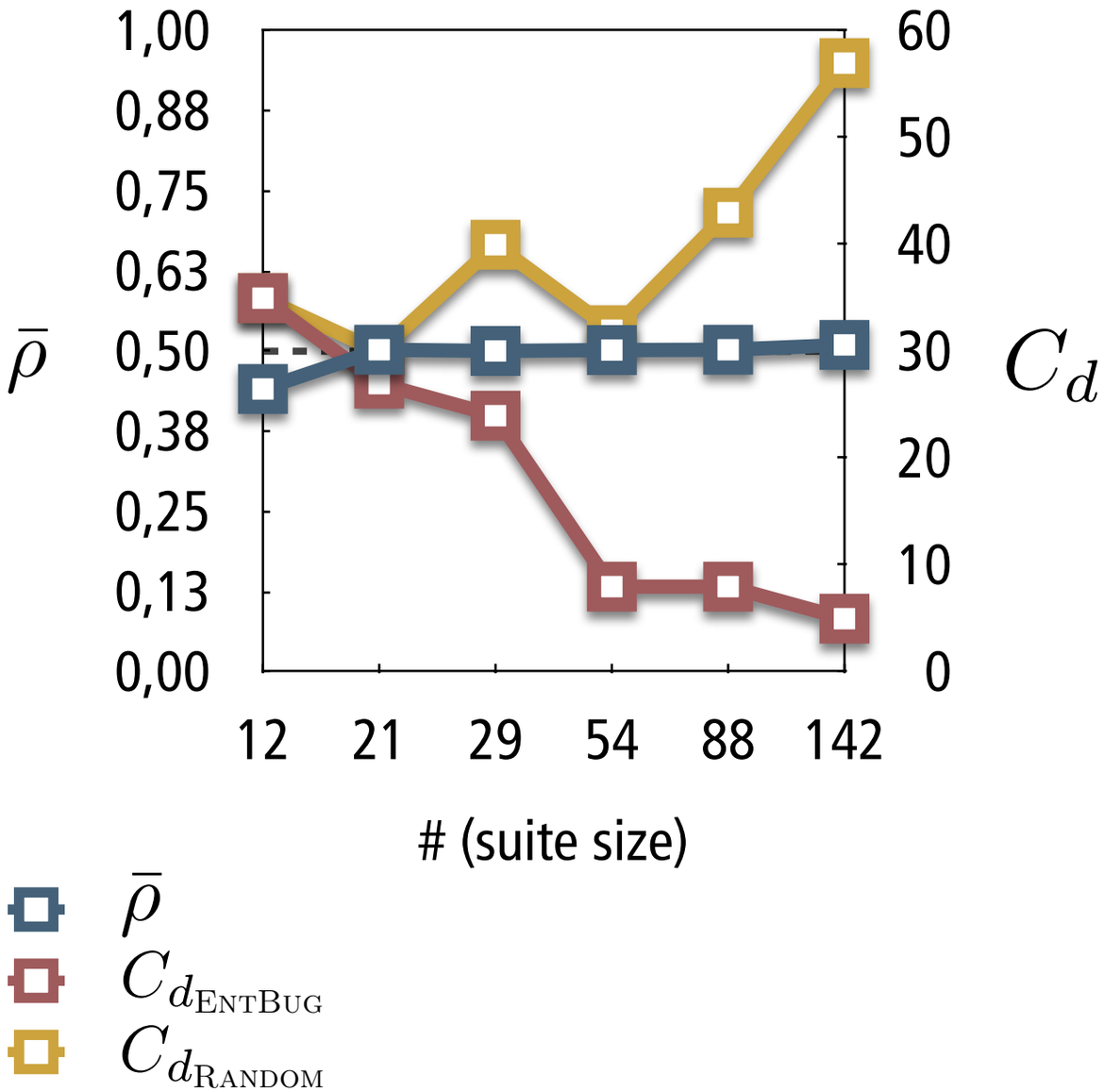




Evaluation - Apache Commons Math #939

```
// org.apache.commons.math3.stat.correlation.Covariance
@@ -279,15 +279,15 @@
  private void checkSufficientData(final RealMatrix matrix) throws
        MathIllegalArgumentException {
    int nRows = matrix.getRowDimension();
    int nCols = matrix.getColumnDimension();
-   if (nRows < 2 || nCols < 2) {
+   if (nRows < 2 || nCols < 1) {
      throw new MathIllegalArgumentException(
        LocalizedFormats.INSUFFICIENT_ROWS_AND_COLUMNS,
          nRows, nCols); ...
```

Evaluation - Apache Commons Math #939



Evaluation - Apache Commons Math #939