UNIVERSITÉ DE RENNES 1 — UMR IRISA — Inria INVENTEURS DU MONDE NUMÉRIQUE — DiverSE Diversity-Centric Software Engineering

Investigate the Matrix: Leveraging Variability to Specialize Software and Test Suites

Paul TEMPLE

December, 7^{th} 2018

Jury:

Myra Cohen, Prof. Iowa State University

Philippe Collet, Prof. Université Nice Sophia Antipolis/ UCA

Yves Le Traon, Prof. Université du Luxembourg

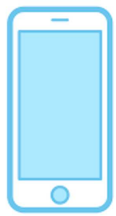
Patrick Pérez, Research Director Valeo.ai

Jean-Marc Jézéquel, Prof. Université de Rennes 1

Mathieu Acher, Mcf. Université de Rennes 1

Modern software

Software is eating the world



Andreessen, Why software is eating the world?, The Wall Street Journal 2011

Capability of being customized

Software Variability by Svahnberg *et al.*

The ability of a software system or artefact to be **efficiently extended, changed, customized or configured** for use in a **particular context**.

Svahnberg *et al.*, A taxonomy of variability realization techniques: Research Articles, Softw. Pract. Exper. 2005

Capability of being customized

Software Variability by Svahnberg *et al.*

The ability of a software system or artefact to be **efficiently extended, changed, customized or configured** for use in a **particular context**.

JHipster: **50** options

Linux Kernel: **15,000** options

$$2^{15,000} \approx 10^{3,250} >> 10^{1,000} >> \text{estimated } \# \text{ of particules}$$

Svahnberg *et al.*, A taxonomy of variability realization techniques: Research Articles, Softw. Pract. Exper. 2005

Adapt to different contexts

options:

- no-mbtree (T or F)
- nr ([100..1000])
- qblur ([0; 1])

$\Rightarrow$

step $= 0.0001$
$\rightarrow$ **18 millions** of configurations

Adapt to different contexts

options:

- no-mbtree (T or F)
- nr ([100..1000])
- qblur ([0; 1])

step = 0.0001
→ **18 millions** of configurations

⇒

Cannot try all configurations

Sampling configurations

- Detected faults depends on the sampling strategy (Medeiros *et al.* ; Sarkar *et al.*)
- Choosing the right sampling strategy is an open problem (Medeiros *et al.*)
- Some configurations may not be valid (Cohen *et al.*, Henard *et al.*, Lamancha *et al.*)

Medeiros *et al.*, A comparison of 10 sampling algorithms for configurable systems, ICSE, 2016

Sarkar *et al.*, Cost-efficient sampling for performance prediction of configurable systems, ASE, 2015

Cohen *et al.*, Constructing Interaction Test Suites for Highly Configurable Systems in the Presence of Constraints: A Greedy approach, IEEE TSE, 2008

Henard *et al.*, Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test configurations for SPL, IEEE TSE, 2014

Lamancha *et al.*, Testing product generation in SPLs using pairwise for

Cannot try all configurations

Predicting performances

- Previously executed configurations are kept into a database (Sincero *et al.*)
- Create a performance-influence model using Machine Learning (Guo *et al.*, Siegmund *et al.*)

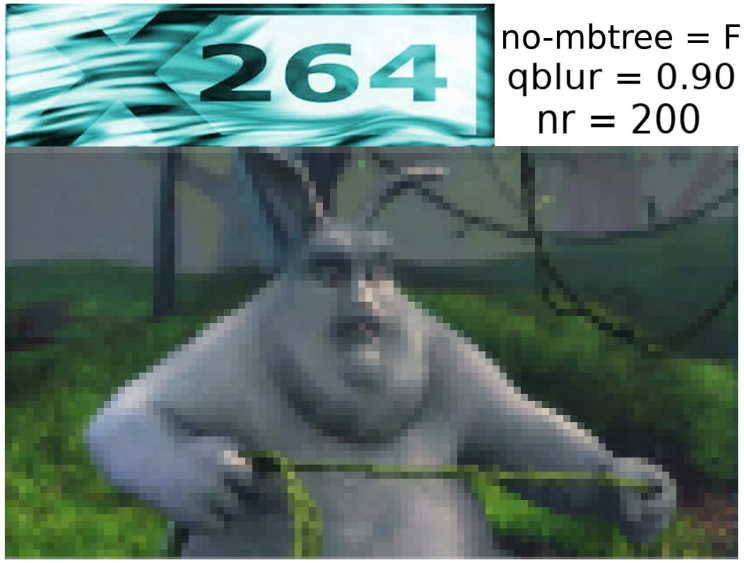
Sincero *et al.*, Approaching non-functional properties of SPLs: Learning from products, APSEC,2010

Siegmund *et al.*,Performance-influence models for highly configurable systems, FSE, 2015

Guo *et al.*, Variability-aware performance prediction: A statistical learning approach, ASE, 2013

Siemgund *et al.*, Scalable prediction of non-functional properties in SPLs: Footprint and memory consumption, Info. and Softw. Technol., 2013

Inputs have an influence

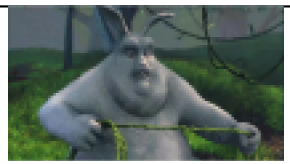
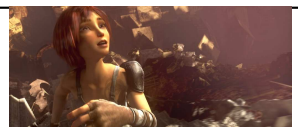


encoding time = 5 min

encoding time = 2 h

encoding time = 10 h

Combining the two

	Program Variants				
Inputs		x264	x264	...	x264
		12	1	...	5
		1	348	...	10
	...				
		50	101	...	260

Problems

- Cartesian product is HUGE
- Testing budget is often limited

$\Rightarrow$ difficult to fill completely the matrix

Contributions

	Program Variants				
Inputs		x264	x264	...	x264
		12	1	...	5
		1	348	...	10
	...				
		50	101	...	260

Automatic Specialization of Software Product Line

Using Machine Learning to Infer constraints for Product Lines, SPLC 2016

VaryLaTeX: Learning Paper Variants That Meet Constraints, VaMoS 2018

Learning Contextual-Variability Model, IEEE Software Vol. 34, Nov. 2017

Test Suite Quality Assessment

Multimorphic Testing, ICSE (Poster), 2018

Empirical Assessment of Multimorphic Testing, Under submission at TSE

Automatic Specialization of Software Product Line

Problem:

- Too many configurations to apply a try-and-error process
- Can we help users by capturing the subset of interesting configurations for the task-at-hand?

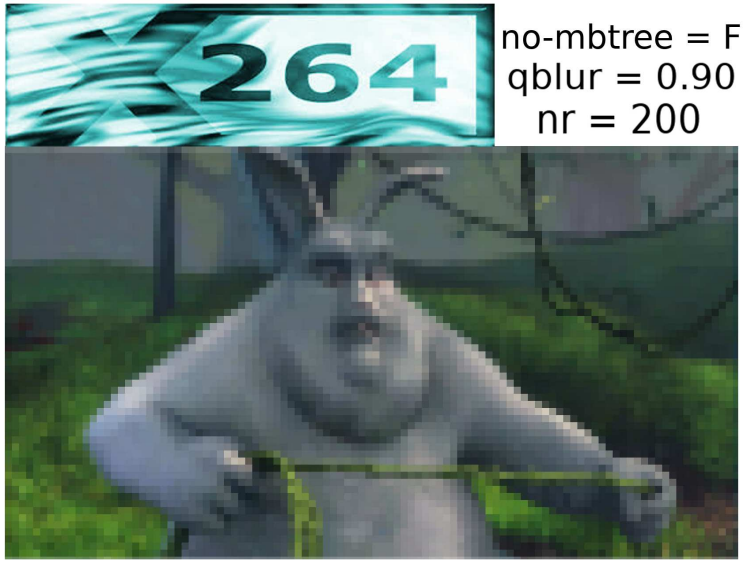
Objective:

- Use Machine Learning techniques to automatically synthesize constraints restraining the space of configurations such that only interesting configurations remain

Results:

- We retrieved constraints that were precise with only few classification errors
- Retrieved constraints were understandable by practitioners

Inputs have an influence



encoding time = 5 min

encoding time = 2 h

encoding time = 10 h

In a different context

Hard to recognize

- Dark, at night
- Unexpected pedestrian crossing the street

$\Rightarrow$ **correct recognition before** the accident

Problem shifting

- **Is it able to** recognize the pedestrian?
- Does it recognize the pedestrian **fast enough**?

"Yes/No" Verdict $\rightarrow$ quality of service assessment
From functional to non-functional perspective

How to build a test suite?

Which one is the best?

		Program Variants		
		Program 1	Program 2	Program 3
Inputs		13	5	7
		100	1500	800
		∞	∞	∞

How to build a test suite?

Which one is the best?

	Program Variants			
		Program 1	Program 2	Program 3
Inputs		13	5	7
		100	1500	800
		∞	∞	∞

Problem

We need to find a measure to rank non-functional tests

How to build a test suite?

Which one is the best?

	Program Variants			
		Program 1	Program 2	Program 3
Inputs		13	5	7
		100	1500	800
		∞	∞	∞

Problem

We need to find a measure to rank non-functional tests

$\Rightarrow$ **several** program variants are needed

Quality of tests

Different techniques:

- Coverage score in structural testing
- Mutation testing

Huang *et al.*, An approach to program testing, ACM Computer Survey, 1975

Andrews *et al.*, Using mutation analysis for assessing and comparing testing coverage criteria, IEEE TSE, 2006

Quality of tests

Different techniques:

- Coverage score in structural testing
- Mutation testing

They are **all** focused on functional properties

Huang *et al.*, An approach to program testing, ACM Computer Survey, 1975

Andrews *et al.*, Using mutation analysis for assessing and comparing testing coverage criteria, IEEE TSE, 2006

How to select tests?

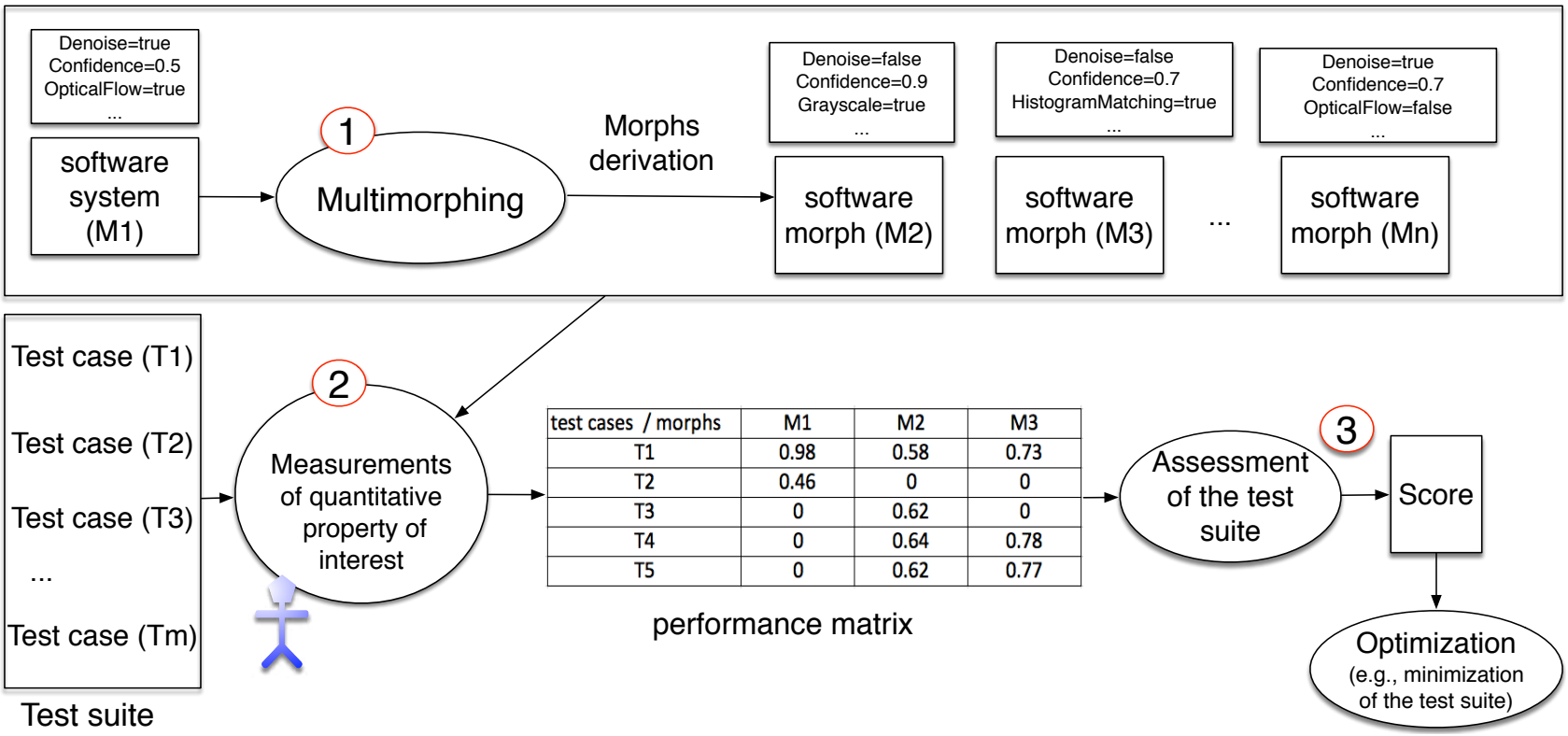
Which test suites are really useful/good?

What does **useful/good** mean?

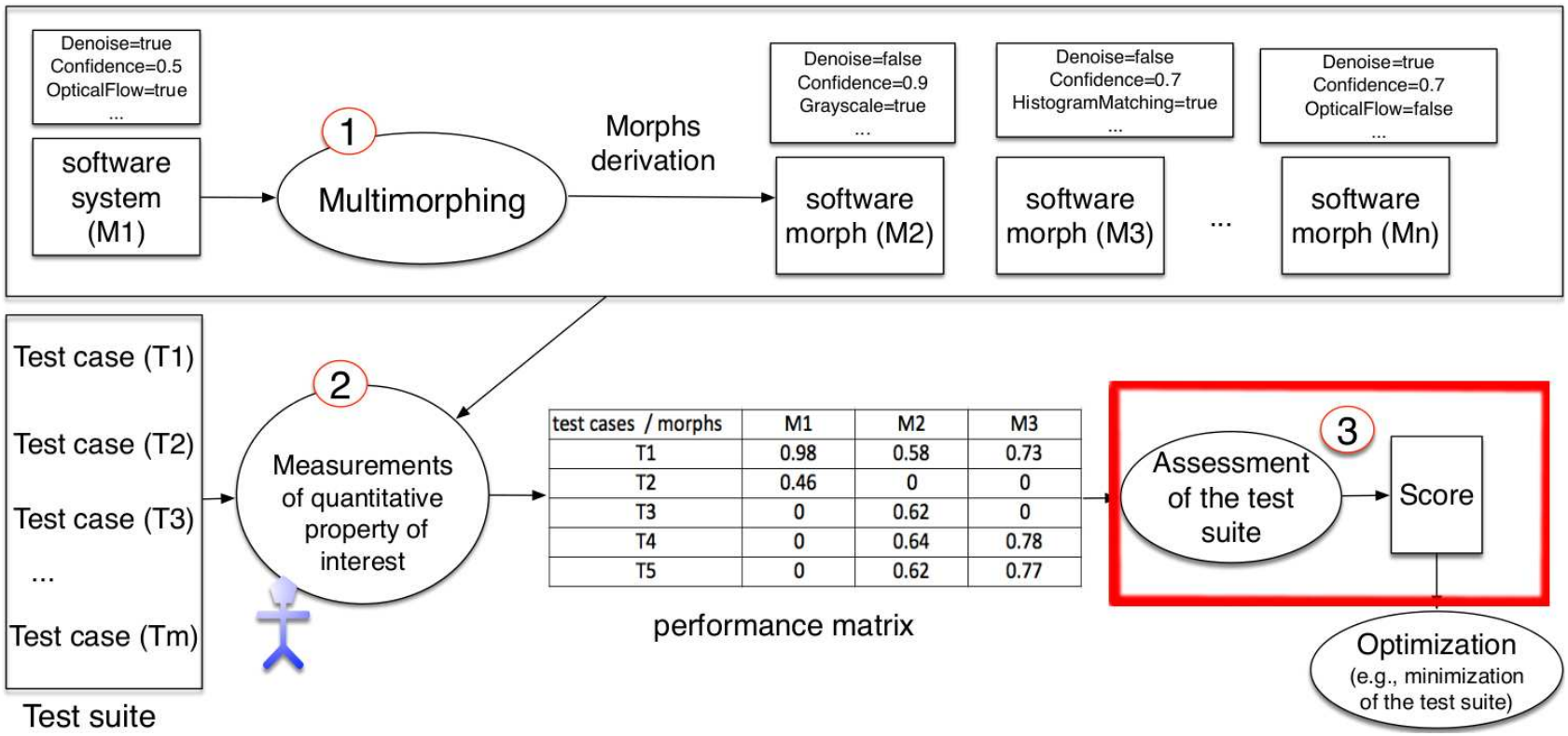
Be aware of the range of performances programs can achieve

A test suite that is able to show significant differences in program variants' performances

Multimorphic Testing



Multimorphic Testing



Desired properties of the score

We want to assign scores to test suites which shows significant differences in the performances:

- P1: Has to be positive
- P2: Considering 2 test suites A and B, with A $\subseteq$ B, score(A) $\leq$ score(B)
- P3: $\forall$ test suites A and B, score($A \cup B$) $\geq$ max(score(A),score(B))

Candidate score

Variance

Used to quantify the dispersion of quantitative measures

Candidate score

Variance

Used to quantify the dispersion of quantitative measures

	Morph 1	Morph 2	Morph 3	Morph 4
Test suite 1	0.2	0.3	0.2	0.4
Test suite 2	0.1	0.1	0.6	0.6

	Test suite 1	Test suite 2	Test suite 1 $\cup$ Test suite 2
Variance	0.009	0.083	0.041

Candidate score

Variance

Used to quantify the dispersion of quantitative measures

	Morph 1	Morph 2	Morph 3	Morph 4
Test suite 1	0.2	0.3	0.2	0.4
Test suite 2	0.1	0.1	0.6	0.6

	Test suite 1	Test suite 2	Test suite 1 $\cup$ Test suite 2
Variance	0.009	0.083	0.041

$$\text{Var.(Test suite 2)} > \text{Var.(Test suite 1} \cup \text{Test suite 2)}$$

P3 is violated

Dispersion score

	Morph 1	Morph 2	Morph 3	Morph 4
Test suite 1	0.98	0.58	0.73	0.65
Test suite 2	0.46	0.2	0.62	0.3

Computation

- Normalize values from the matrix in $[0; 1]$
- Divide $[0; 1]$ into equally distributed bins (# of morphs)
- Bins for which values fall in their range are activated
- Count the number of activated bins
- Divide by the number of bins

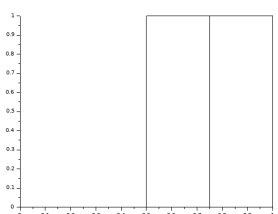
Dispersion score

	Morph 1	Morph 2	Morph 3	Morph 4
Test suite 1	0.98	0.58	0.73	0.65
Test suite 2	0.46	0.2	0.62	0.3

Computation

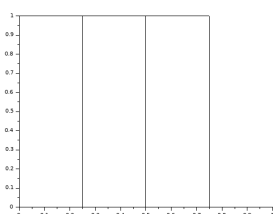
- Normalize values from the matrix in $[0; 1]$
- Divide $[0; 1]$ into equally distributed bins (# of morphs)
- Bins for which values fall in their range are activated
- Count the number of activated bins
- Divide by the number of bins

T1:



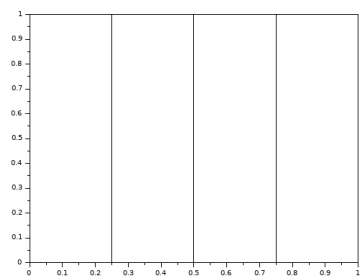
$= \frac{2}{4}$

T2:



$= \frac{3}{4}$

$T1 \cup T2$:



$= \frac{4}{4}$

Research Questions

Is the measure right?

- Does the dispersion score fulfill the desired properties?
- Are different dispersion scores assigned to test suites according to their ability to exhibit different performances?
- Is dispersion score sensitive to the selection of morphs?

Is it a right measure?

- Is there a correlation between the actual (relative) effectiveness of test suites and their dispersion score?

Evaluation

3 cases:

Case	App. Domain	# morphs	# test suites
OpenCV	Tracking in videos	252	49
COCO	Obj. rec. in images	52	12
Haxe	Code generation	21	84

Does the dispersion score fulfill the desired properties?

P1: score has to be positive

- It is a quotient of positive values $\rightarrow [0; 1]$

Does the dispersion score fulfill the desired properties?

P1: score has to be positive

- It is a quotient of positive values $\rightarrow [0; 1]$

P2: if A $\subseteq$ B, score(A) $\leq$ score(B)

- If A $\subseteq$ B, # activated_bins(A) $\leq$ # activated_bins(B)
- Score(A) $\leq$ score(B)

Does the dispersion score fulfill the desired properties?

P1: score has to be positive

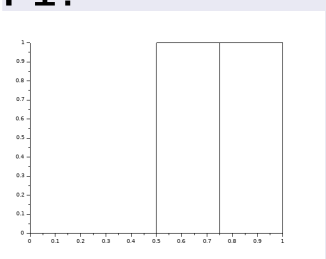
- It is a quotient of positive values $\rightarrow [0; 1]$

P2: if A $\subseteq$ B, score(A) $\leq$ score(B)

- If A $\subseteq$ B, # activated_bins(A) $\leq$ # activated_bins(B)
- Score(A) $\leq$ score(B)

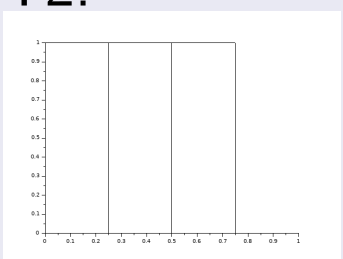
P3: score($A \cup B$) $\geq$ max(score(A),score(B))

T1:



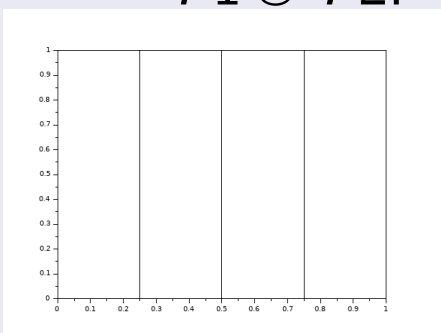
$= \frac{2}{4}$

T2:



$= \frac{3}{4}$

$T1 \cup T2$:



$= \frac{4}{4}$

Evaluation

Is the measure right?

- Are different dispersion scores assigned to test suites according to their ability to exhibit different performances?
- Is dispersion score sensitive to the selection of morph?

Evaluation: Discriminative power of the dispersion score?

Are test suites assigned with different scores?

- OpenCV: $[0.08; 0.207]$
- COCO: $[0.308; 0.423]$
- Haxe: $[0.047; 0.143]$

$\Rightarrow$ Test suites do not have the same dispersion scores

Evaluation: Discriminative power of the dispersion score?

Are test suites assigned with different scores?

- OpenCV: $[0.08; 0.207]$
- COCO: $[0.308; 0.423]$
- Haxe: $[0.047; 0.143]$

$\Rightarrow$ Test suites do not have the same dispersion scores

Scores depend on a set of morphs $\Rightarrow$ the absolute values are meaningless

Evaluation: Sensitivity analysis

Sensitive to selection of morphs?

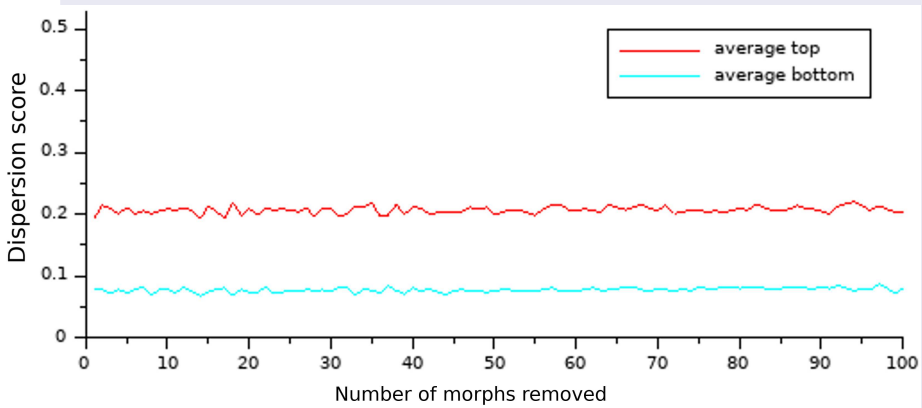
- Remove up to half the morphs
- Each time one morph is removed; assess the dispersion scores
- Repeat morph removal 50 times

Evaluation: Sensitivity analysis

Sensitive to selection of morphs?

- Remove up to half the morphs
- Each time one morph is removed; assess the dispersion scores
- Repeat morph removal 50 times

OpenCV



- Dispersion scores **remain stable**
- Adding or not one morph **should not be critical**

Is it a right measure?

Is there a correlation between the actual (relative) effectiveness of test suites and their dispersion score?

Need to build larger test suites → aggregate test suites

- Criterion of maximization: **Maximize the number of activated bins**
- Maximizing the score $\neq$ taking the n top individual test suites
- Exhaustive search of the best combination

Evaluation: Dispersion score and effectiveness of test suites

Correlation between dispersion scores and effectiveness of test suites?

- OpenCV: can we tell apart good object recognition algorithms from bad ones?
- COCO: can we keep a similar ranking while reducing the size of the benchmark?
- Haxe: can we find bugs with our test suite?

Evaluation: Dispersion score and effectiveness of test suites

Correlation between dispersion scores and effectiveness of test suites?

- **OpenCV: can we tell apart good object recognition algorithms from bad ones?**

- We compute a new test suite maximizing our criterion (n=5 out of 49)
- Experts took 12 morphs
- 6 supposed to perform poorly
- 6 supposed to perform well

→ Can we tell them apart with our test suite?

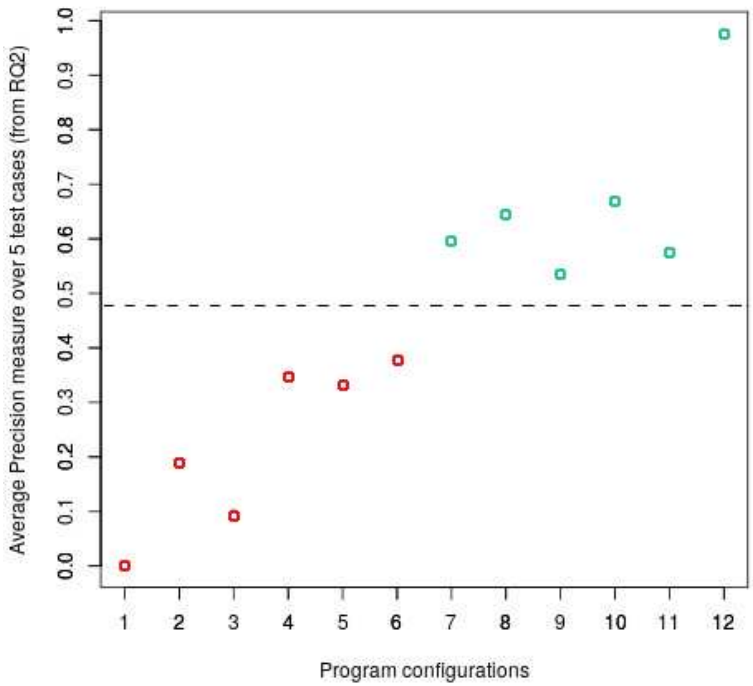
Evaluation: Dispersion score and effectiveness of test suites

Correlation between dispersion scores and effectiveness of test suites?

- **OpenCV: can we tell apart good object recognition algorithms from bad ones?**

- We compute a new test suite maximizing our criterion (n=5 out of 49)
- Experts took 12 morphs
- 6 supposed to perform poorly
- 6 supposed to perform well

→ Can we tell them apart with our test suite?



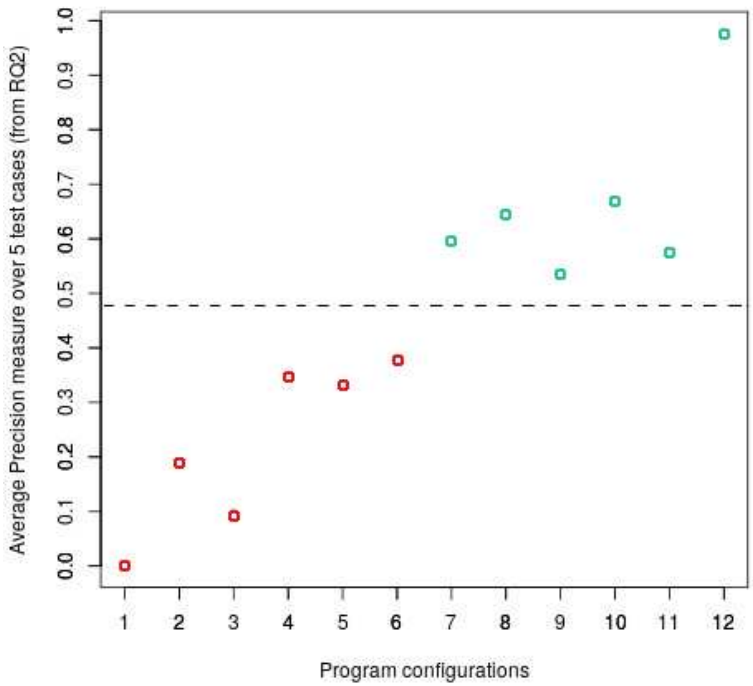
Evaluation: Dispersion score and effectiveness of test suites

Correlation between dispersion scores and effectiveness of test suites?

- **OpenCV: can we tell apart good object recognition algorithms from bad ones?**

If 5 test suites are taken randomly?

- Repeat 10 times
- **Average**: 4 morphs misclassified
- **Best case**: 2 morphs misclassified



Evaluation: Dispersion score and effectiveness of test suites

RQ2

- **COCO: can we keep a similar ranking while reducing the size of the test suite?**

- 12 test suites $\rightarrow$ 40k images
- We compute a new test suite maximizing our criterion (n=5)
- Rank competitors again
- Correlation between the two rankings: Spearman correlation coefficient: 0.998

$\Rightarrow$ We can keep a similar ranking with a smaller test suite

---

Spearman, The proof and measurement of association between two things, American Journal of Psychology, 1904

Evaluation: Dispersion score and effectiveness of test suites

RQ2

- **Haxe: can we find bugs with our test suite?**

- 1 bug found with the original test suite
- PHP generator did not use the right data structure

Boussaa *et al.*, Automatic non-functional testing of code generators families, 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences 2016

Evaluation: Dispersion score and effectiveness of test suites

RQ2

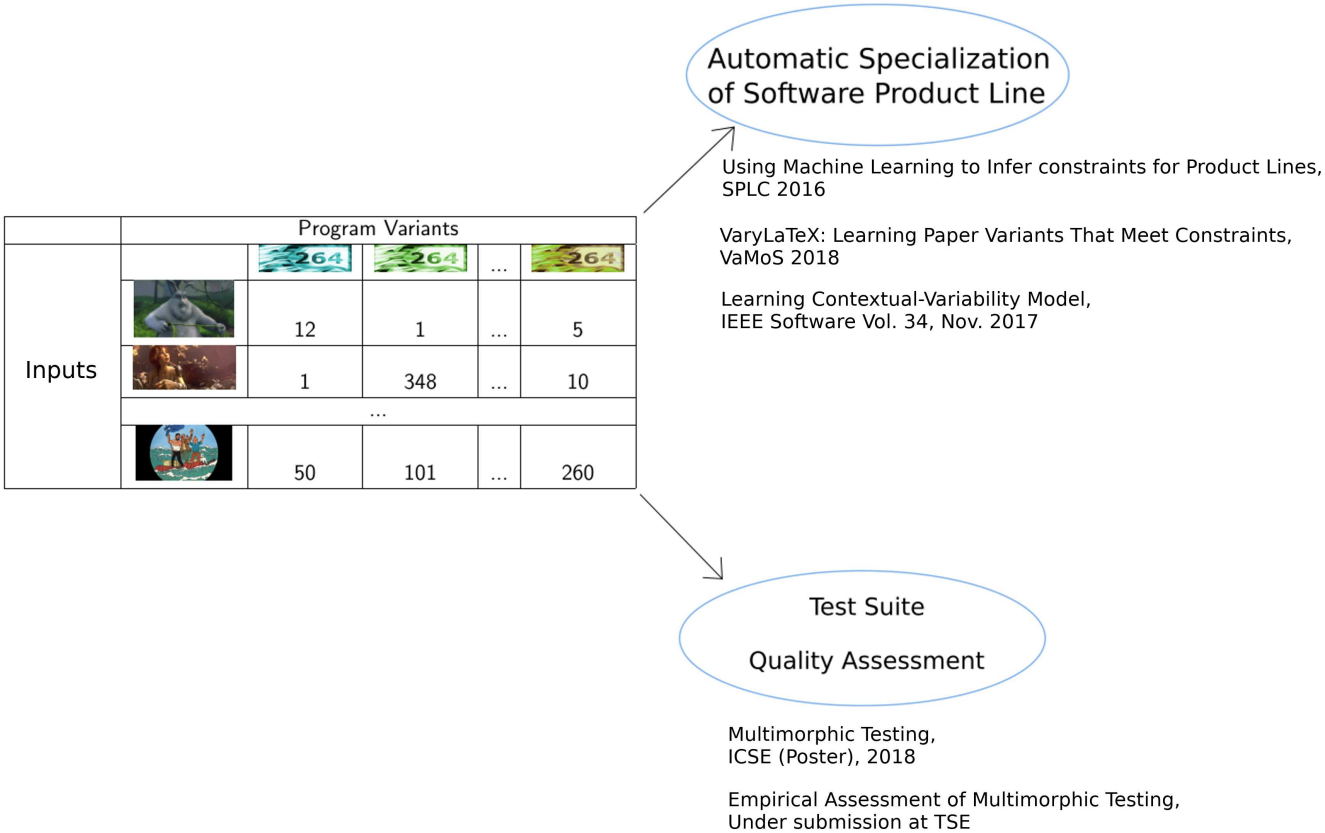
- **Haxe: can we find bugs with our test suite?**

- 1 bug found with the original test suite
- PHP generator did not use the right data structure
- We compute a new test suite maximizing our criterion (n=5 out of 84)
- We are able to find this bug again

$\Rightarrow$ Testing effort drastically reduced

Boussaa *et al.*, Automatic non-functional testing of code generators families, 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences 2016

Conclusion



2 major contributions

Shrinking the size of the matrix along software and test suite dimension

Perspectives

Multimorphic testing

- Dispersion score is one solution, can we find others?
- Exhaustive search is costly, find an other way to combine test suites?
- How to combine Multimorphic testing with other testing techniques?
- Is it a first move to test Machine Learning based systems?

Perspectives

Multimorphic testing

- Dispersion score is one solution, can we find others?
- Exhaustive search is costly, find an other way to combine test suites?
- How to combine Multimorphic testing with other testing techniques?
- Is it a first move to test Machine Learning based systems?

Automatic Specialization

- Which Machine Learning technique to use such that it is powerful while maintaining constraint's understandability?
- Are Adversarial Machine Learning techniques useful in this context?

Thank you for listening