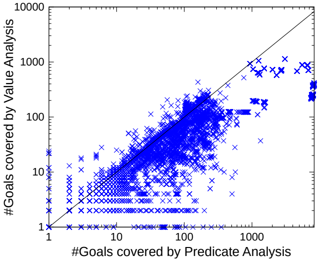
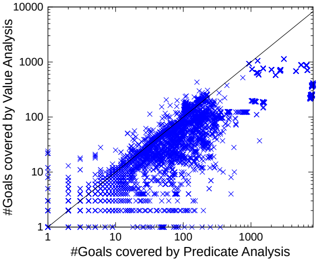
Towards Algorithm Selection for Test-Case Generation with CPAchecker

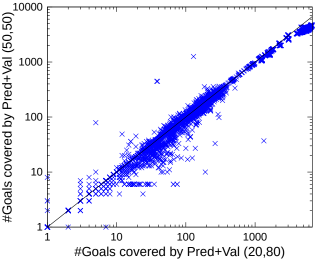
Marie-Christine Jakobs

Why Algorithm Selection?

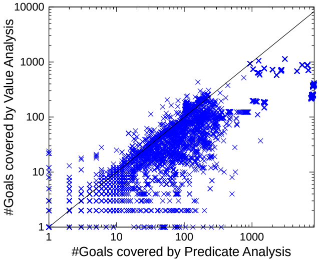


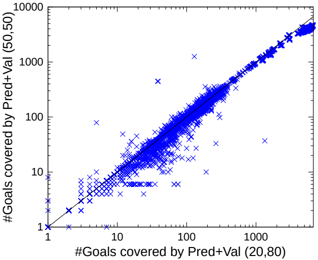
Why Algorithm Selection?





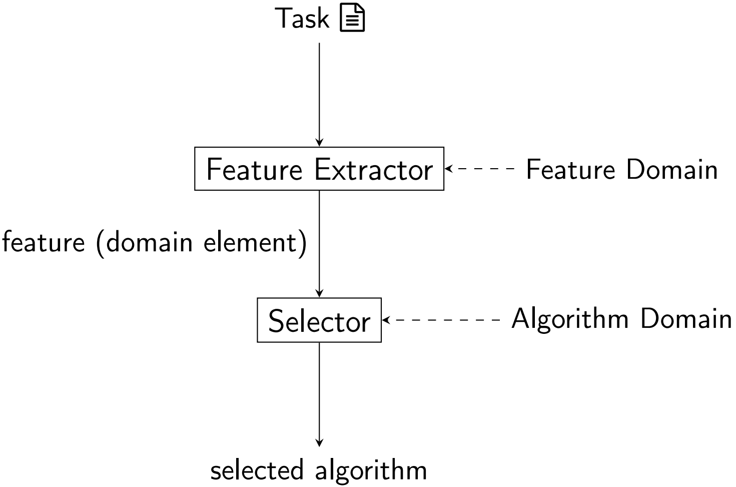
Why Algorithm Selection?





No analysis configuration superior

The Principle of Algorithm Selection



Our Feature Domain

Boolean Features

already used for selection of verifier configurations

D. Beyer, M. Dangl: Strategy Selection for Software Verification Based on Boolean Features - A Simple but Effective Approach, ISoLA, 2018.

Our Feature Domain

Boolean Features

hasArray ([]) if the test task (program) uses a variable with an array type

already used for selection of verifier configurations

D. Beyer, M. Dangl: Strategy Selection for Software Verification Based on Boolean Features - A Simple but Effective Approach, ISoLA, 2018.

Our Feature Domain

Boolean Features

hasArray ([]) if the test task (program) uses a variable with an array type

hasComposite (○) if the test task (program) uses a variable with a struct or union type

already used for selection of verifier configurations

D. Beyer, M. Dangl: Strategy Selection for Software Verification Based on Boolean Features - A Simple but Effective Approach, ISoLA, 2018.

Our Feature Domain

Boolean Features

hasArray ([]) if the test task (program) uses a variable with an array type

hasComposite (○) if the test task (program) uses a variable with a struct or union type

hasFloat ($\mathbb{R}$) if the test task (program) uses a variable of type float, double, or long double

already used for selection of verifier configurations

D. Beyer, M. Dangl: Strategy Selection for Software Verification Based on Boolean Features - A Simple but Effective Approach, ISoLA, 2018.

Our Feature Domain

Boolean Features

hasArray ([]) if the test task (program) uses a variable with an array type

hasComposite (○) if the test task (program) uses a variable with a struct or union type

hasFloat (ℝ) if the test task (program) uses a variable of type float, double, or long double

hasLoop (⥁) if the test task (program) has a loop

already used for selection of verifier configurations

D. Beyer, M. Dangl: Strategy Selection for Software Verification Based on Boolean Features - A Simple but Effective Approach, ISoLA, 2018.

What to Select? – Our Selection Instances

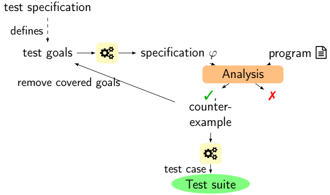
Analysis Algorithm

Time Limits for CoVeriTest

What to Select? – Our Selection Instances

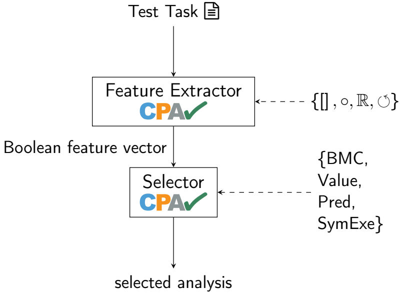
Analysis Algorithm

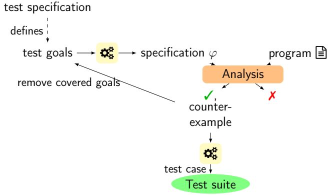
Time Limits for CoVeriTest



What to Select? – Our Selection Instances

Analysis Algorithm

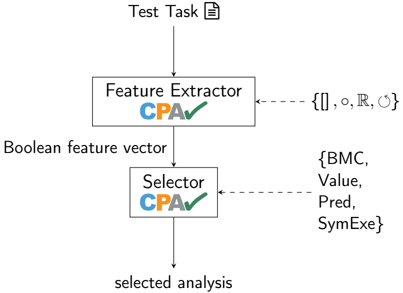




Time Limits for CoVeriTest

What to Select? – Our Selection Instances

Analysis Algorithm



Test Task

Feature Extractor CPA✓

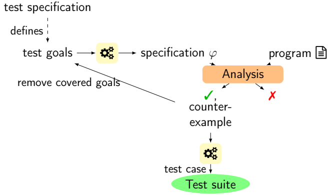
$\{[\,], \circ, \mathbb{R}, \circlearrowleft\}$

Boolean feature vector

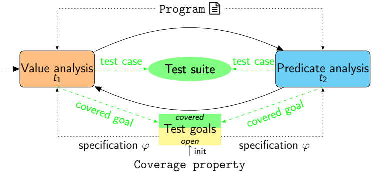
Selector CPA✓

{BMC, Value, Pred, SymExe}

selected analysis

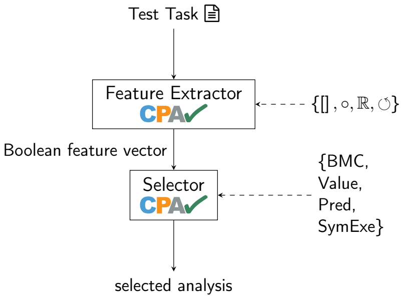


Time Limits for CoVeriTest



What to Select? – Our Selection Instances

Analysis Algorithm


Test Task

Feature Extractor CPA✓

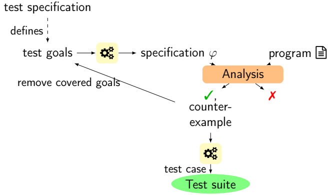
$\{[\,], \circ, \mathbb{R}, \circlearrowleft\}$

Boolean feature vector

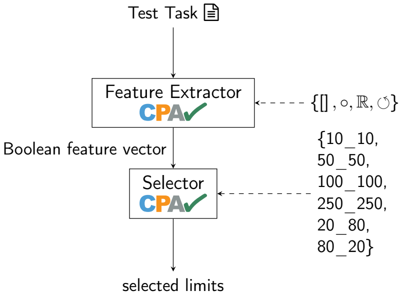
Selector CPA✓

{BMC, Value, Pred, SymExe}

selected analysis



Time Limits for CoVeriTest


Test Task

Feature Extractor CPA✓

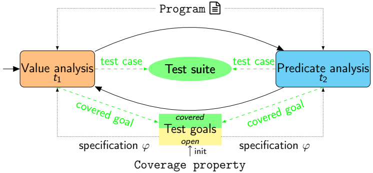
$\{[\,], \circ, \mathbb{R}, \circlearrowleft\}$

Boolean feature vector

Selector CPA✓

{10_10, 50_50, 100_100, 250_250, 20_80, 80_20}

selected limits



The Basis for our Selectors

Test Tasks 7 644 programs considered in SV-COMP 2019
branch coverage
(assumes in CPAchecker's representation)

Configurations
- BMC, Predicate, SymExe, Value
- CoVeriTest with Predicate+Value, reuse-arg
  all time limits from STTT evaluation

Experimental Data Coverage data from STTT artifact
extended with computed feature
(additional experiments)

D. Beyer, M.-C. Jakobs: Cooperative verifier-based testing with CoVeriTest. STTT 23(3), 2021.

Towards a Selector for Test-Case Generation Algorithms

Features							
[]	○	ℝ	↻	BMC	Predicate	SymExe	Value
✗	✗	✗	✗	167	201	202	183
✗	✗	✗	✓	1047	2423	1728	1284
✗	✗	✓	✗	116	115	72	69
✗	✗	✓	✓	47	15	20	25
✗	✓	✗	✗	26	41	24	23
✗	✓	✗	✓	168	265	141	100
✗	✓	✓	✗	63	64	56	49
✗	✓	✓	✓	21	21	24	16
✓	✗	✗	✗	18	6	6	6
✓	✗	✗	✓	452	1092	514	412
✓	✗	✓	✗	0	4	0	0
✓	✗	✓	✓	17	3	11	10
✓	✓	✗	✗	10	14	13	13
✓	✓	✗	✓	611	1713	501	365
✓	✓	✓	✗	24	51	21	18
✓	✓	✓	✓	37	38	38	25

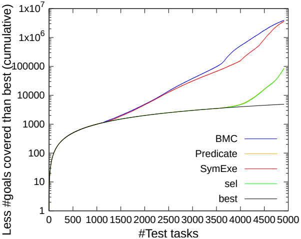
- Value analysis is outperformed
- Predicate analysis often best

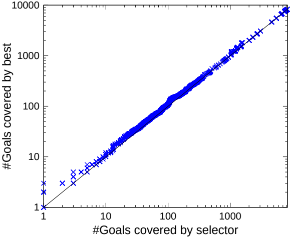
A Selector for or Test-Case Generation Algorithms

Selector

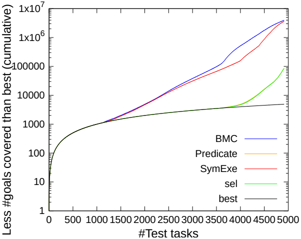
$$
\mathit{sel}_{\mathrm{analysis}}(\mathit{hasArray}, \mathit{hasComposite}, \mathit{hasFloat}, \mathit{hasLoop}) :=
\begin{cases}
\mathit{BMC} & \text{if } \begin{array}{ll} & \neg\mathit{hasComposite} \land \mathit{hasLoop} \land \mathit{hasFloat} \\ \lor & \neg\mathit{hasComposite} \land \neg\mathit{hasLoop} \land (\mathit{hasArray} \oplus \mathit{hasFloat}) \end{array} \\
\mathit{SymExe} & \text{if } \neg\mathit{hasArray} \land \left(\begin{array}{ll} & \mathit{hasComposite} \land \mathit{hasFloat} \land \mathit{hasLoop} \\ \lor & \neg\mathit{hasComposite} \land \neg\mathit{hasFloat} \land \neg\mathit{hasLoop} \end{array} \right) \\
\mathit{Predicate} & \text{else}
\end{cases}
$$

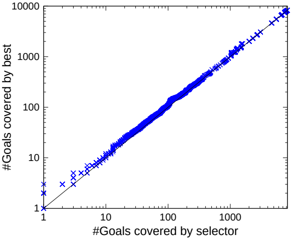
Performance of Selector for Test-Case Generation Algorithms on Existing Data

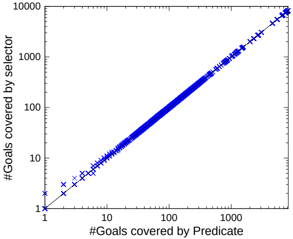




Performance of Selector for Test-Case Generation Algorithms on Existing Data

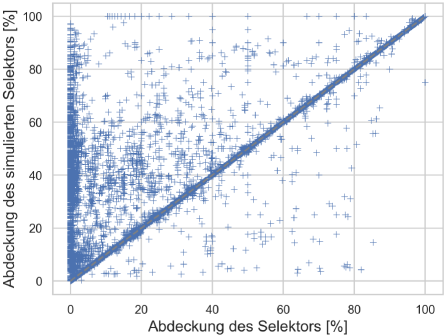






Transferability to new Benchmark Set

15 672 programs of SV-COMP 2024



Selector does not generalize well

Towards a Selector for Time Limits in CoVeriTest

Features									
[]	○	ℝ	↺	100_100	10_10	20_80	250_250	50_50	80_20
✗	✗	✗	✗	237	237	237	237	237	237
✗	✗	✗	✓	1772	1760	2830	1772	1841	1751
✗	✗	✓	✗	129	129	129	129	129	129
✗	✗	✓	✓	54	49	50	54	54	54
✗	✓	✗	✗	41	41	41	41	41	41
✗	✓	✗	✓	309	303	307	314	308	307
✗	✓	✓	✗	102	102	102	100	102	102
✗	✓	✓	✓	42	42	42	42	42	42
✓	✗	✗	✗	5	10	12	5	5	5
✓	✗	✗	✓	1011	1084	1173	995	1033	979
✓	✗	✓	✗	0	4	0	0	0	0
✓	✗	✓	✓	15	17	15	15	15	15
✓	✓	✗	✗	14	14	14	14	14	14
✓	✓	✗	✓	1013	1062	1658	985	959	822
✓	✓	✓	✗	59	59	59	53	59	59
✓	✓	✓	✓	69	70	80	64	70	67

- No difference for some features

Towards a Selector for Time Limits in CoVeriTest

Features									
[]	○	ℝ	↺	100_100	10_10	20_80	250_250	50_50	80_20
✗	✗	✗	✗	237	237	237	237	237	237
✗	✗	✗	✓	1772	1760	2830	1772	1841	1751
✗	✗	✓	✗	129	129	129	129	129	129
✗	✗	✓	✓	54	49	50	54	54	54
✗	✓	✗	✗	41	41	41	41	41	41
✗	✓	✗	✓	309	303	307	314	308	307
✗	✓	✓	✗	102	102	102	100	102	102
✗	✓	✓	✓	42	42	42	42	42	42
✓	✗	✗	✗	5	10	12	5	5	5
✓	✗	✗	✓	1011	1084	1173	995	1033	979
✓	✗	✓	✗	0	4	0	0	0	0
✓	✗	✓	✓	15	17	15	15	15	15
✓	✓	✗	✗	14	14	14	14	14	14
✓	✓	✗	✓	1013	1062	1658	985	959	822
✓	✓	✓	✗	59	59	59	53	59	59
✓	✓	✓	✓	69	70	80	64	70	67

- No difference for some features
- 20_80 limit often best

Towards a Selector for Time Limits in CoVeriTest

Features									
[]	○	ℝ	↻	100_100	10_10	20_80	250_250	50_50	80_20
✗	✗	✗	✗	237	237	237	237	237	237
✗	✗	✗	✓	1772	1760	2830	1772	1841	1751
✗	✗	✓	✗	129	129	129	129	129	129
✗	✗	✓	✓	54	49	50	54	54	54
✗	✓	✗	✗	41	41	41	41	41	41
✗	✓	✗	✓	309	303	307	314	308	307
✗	✓	✓	✗	102	102	102	100	102	102
✗	✓	✓	✓	42	42	42	42	42	42
✓	✗	✗	✗	5	10	12	5	5	5
✓	✗	✗	✓	1011	1084	1173	995	1033	979
✓	✗	✓	✗	0	4	0	0	0	0
✓	✗	✓	✓	15	17	15	15	15	15
✓	✓	✗	✗	14	14	14	14	14	14
✓	✓	✗	✓	1013	1062	1658	985	959	822
✓	✓	✓	✗	59	59	59	53	59	59
✓	✓	✓	✓	69	70	80	64	70	67

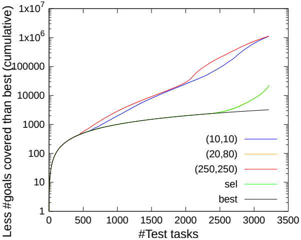
- No difference for some features
- 20_80 limit often best
- Limits 10_10, 20_80, 250_250 sufficient

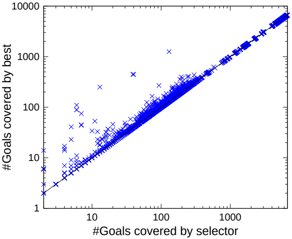
A Selector for Time Limits in CoVeriTest

Selector

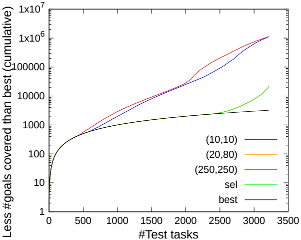
$$
\mathit{sel}_{\mathrm{limit}}(\mathit{hasArray}, \mathit{hasComposite}, \mathit{hasFloat}, \mathit{hasLoop}) :=
\begin{cases}
250\_250 & \text{if } \neg\mathit{hasArray} \wedge (\mathit{hasComposite} \oplus \mathit{hasFloat}) \\
10\_10 & \text{if } \mathit{hasArray} \wedge \neg\mathit{hasComposite} \wedge \mathit{hasFloat} \\
20\_80 & \text{else}
\end{cases}
$$

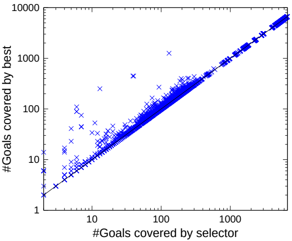
Performance of Selector for Time Limits on Existing Data

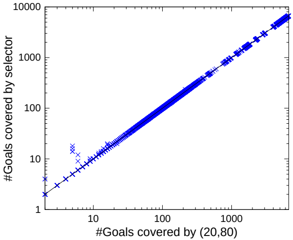




Performance of Selector for Time Limits on Existing Data







Critical Reflection

- Suitability of Data Set
  - Observed lack of generalizability of designed selector
  - Imbalance

Features				# tasks		
[]	○	ℝ	↺	total	Δ coverage (ana)	Δ coverage (limits)
✗	✗	✗	✗	241	83	4
✗	✗	✗	✓	2969	2209	1253
✗	✗	✓	✗	129	70	0
✗	✗	✓	✓	54	51	5
✗	✓	✗	✗	41	19	0
✗	✓	✗	✓	323	245	25
✗	✓	✓	✗	106	89	6
✗	✓	✓	✓	42	41	0
✓	✗	✗	✗	18	12	13
✓	✗	✗	✓	1267	1051	335
✓	✗	✓	✗	4	4	4
✓	✗	✓	✓	17	15	2
✓	✓	✗	✗	14	4	0
✓	✓	✗	✓	2276	2192	1666
✓	✓	✓	✗	67	63	14
✓	✓	✓	✓	80	75	17

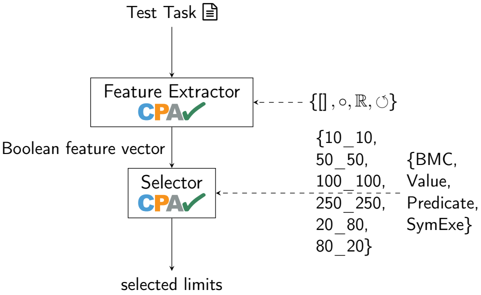
Critical Reflection (cont.)

- Suitability of Feature Domain

Features							
[]	○	ℝ	↻	BMC	Predicate	SymExe	Value
✗	✗	✗	✗	0	24	17	5
✗	✗	✗	✓	45	1036	257	20
✗	✗	✓	✗	4	10	0	0
✗	✗	✓	✓	17	0	1	0
✗	✓	✗	✗	0	13	0	0
✗	✓	✗	✓	31	114	4	0
✗	✓	✓	✗	5	39	0	0
✗	✓	✓	✓	1	15	2	0
✓	✗	✗	✗	12	0	0	0
✓	✗	✗	✓	77	551	26	5
✓	✗	✓	✗	0	4	0	0
✓	✗	✓	✓	5	0	0	0
✓	✓	✗	✗	0	1	0	0
✓	✓	✗	✓	254	785	71	60
✓	✓	✓	✗	0	34	0	0
✓	✓	✓	✓	10	30	11	51

Features									
[]	○	ℝ	↻	100_100	10_10	280_80	250_250	50_50	80_20
✗	✗	✗	✗	0	0	0	0	0	0
✗	✗	✗	✓	1	12	1076	3	76	3
✗	✗	✓	✗	0	0	0	0	0	0
✗	✗	✓	✓	0	0	0	0	0	0
✗	✓	✗	✗	0	0	0	0	0	4
✗	✓	✗	✓	0	0	0	0	0	0
✗	✓	✓	✗	1	3	2	5	2	0
✗	✓	✓	✓	0	0	0	0	0	0
✓	✗	✗	✗	0	5	7	0	0	0
✓	✗	✗	✓	4	32	101	18	5	7
✓	✗	✓	✗	0	4	0	0	0	0
✓	✗	✓	✓	0	2	0	0	0	0
✓	✓	✗	✗	0	0	0	0	0	0
✓	✓	✗	✓	89	116	627	118	50	91
✓	✓	✓	✗	0	0	0	0	0	0
✓	✓	✓	✓	0	0	9	0	0	0

Conclusion



Test Task

Feature Extractor

CPA✓

$\{[\,], \circ, \mathbb{R}, \circlearrowleft\}$

Boolean feature vector

Selector

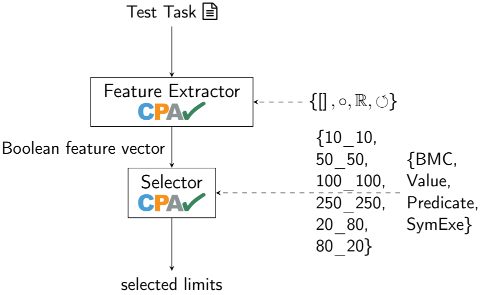
CPA✓

{10_10,
50_50,
100_100,
250_250,
20_80,
80_20}

{BMC,
Value,
Predicate,
SymExe}

selected limits

Conclusion



Test Task

Feature Extractor

CPA✓

$\{[]\,, \circ, \mathbb{R}, \circlearrowleft\}$

Boolean feature vector

Selector

CPA✓

{10_10,
50_50,
100_100,
250_250,
20_80,
80_20}

{BMC,
Value,
Predicate,
SymExe}

selected limits

Lessons learnt

- Boolean features not as promising as in verification
- Generalizability of selector questionable
- Suitability of data needs to be considered