Adaptive testing of dynamic systems

Antonia Bertolino, Breno Miranda   *Roberto Pietrantuono, Stefano Russo*

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

***Dessert* Research Group**
Università di Napoli Federico II

DEpendable Systems
Software Engineering
DESSERT
Research Team

GAUSS Project - Milano, February 2017

Collaboration CNR-UNINA

Joint activity on WP6 – online V&V

- Research area: **reliability testing**
  - Testing for improvement of *delivered reliability*
- First output: a technique for adaptive test selection - ***covrel***
  - Will be presented in May at ICSE 2017

The COVREL approach

Goal

- Test case *selection* for reliability improvement

Idea

- Combine **coverage**-driven and **operational** testing
- ***Adaptive*** selection

Method

- **Iterative test allocation** to partitions driven by online test results
- Test selection within partition based on **count spectrum**

Evaluation

- Experiment on 18 version from 4 subjects from SIR
- Comparison against *operational testing* and *white-box testing*
- *A **prototype** is available for repeatability:* http://labsedc.isti.cnr.it/covrel2017

Background

- **Operational profile testing** is a pillar of SRE
  - Selecting inputs w.r.t. the expected usage at runtime
  - Suitable for *improvement* or *assessment* of reliability
    - **PROS**
      - The most natural way to deal with *reliability = p(failure in operation)*
    - **CONS**
      - "Saturation" effect at high reliability levels
      - Profile knowledge

Background

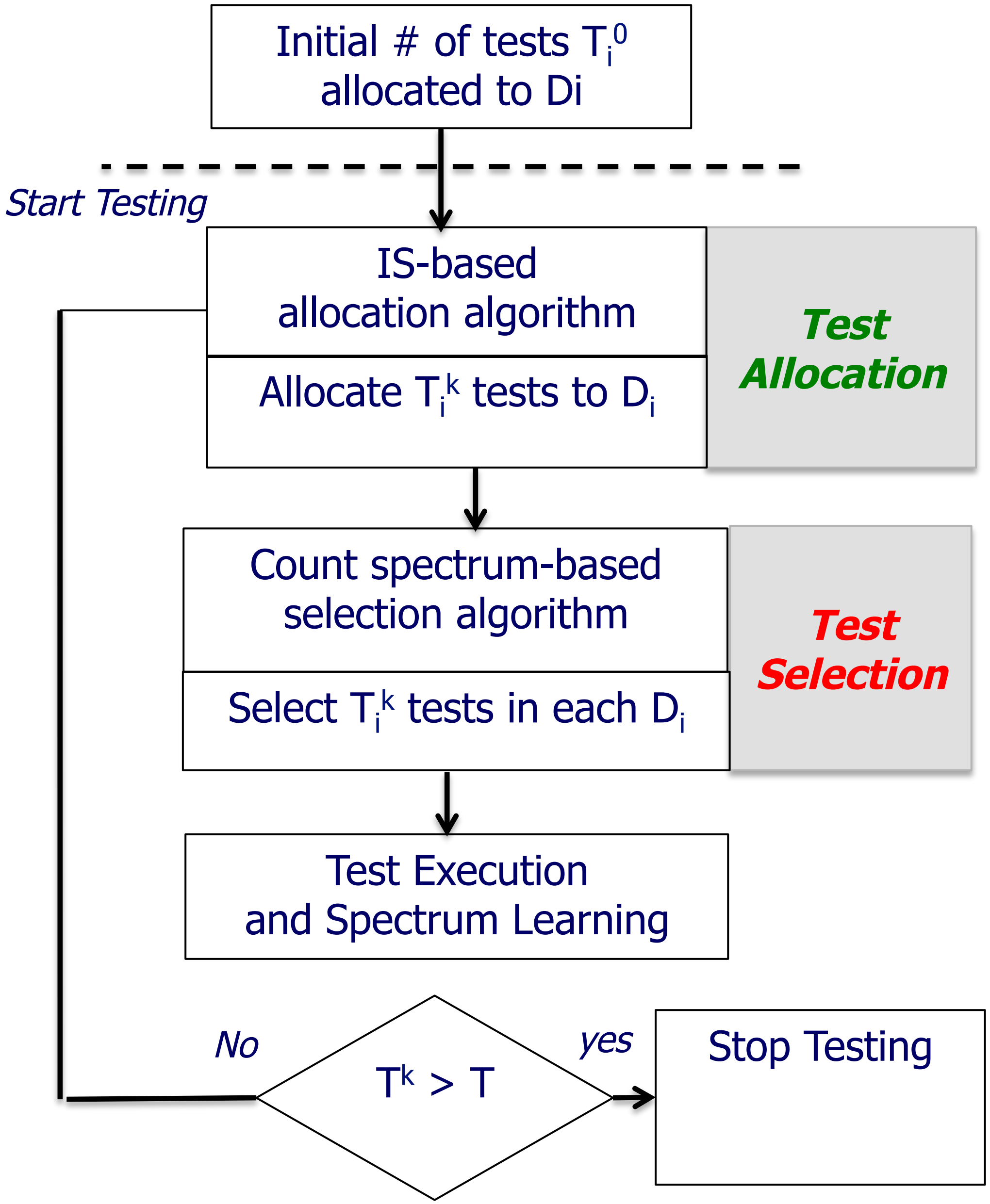
- White-box testing based on **count spectra**
  - A spectrum characterizes a program's behavior by recording the set of entities that are exercised as the program executes.
  - Traditional coverage-based criteria are based on “hit-spectrum”, i.e. they count if an entity is covered (1) or not (0)
  - We use count spectra (record also the number of times an entity is executed), with the aim of considering how frequently entities are exercised

- **IDEA**
  - Considering the frequency of coverage can help bias test selection according to a user’s profile
  - Entities can be weighted proportionally (as in AST’16), or vice versa inversely proportional (as in *covrel*), to usage frequency

Branch ID	Hit	Count
1	1	427
2	1	10834
3	1	11623
4	0	0
5	1	487
6	1	3972
7	1	10543
8	1	87
9	0	0
10	1	67

Overview

- **Notation**
  - D_i: subdomains (partitions)
  - T: budget (test cases)
  - T_i: tests allocated to D_i
  - Profile: distribution over D_i
  - p_i: prob. of taking inputs from D_i
  - k: iteration index
- **Assumptions**
  - Perfect oracle
  - Independence runs of TC
  - Output independent of the history
  - Coverage associated with TCs can be obtained



Test cases allocation

Test Allocation

- Adaptive re-allocation
  - **Allocation objective:** direct more tests where actually needed
    - φ_i : failure rate = number of failing tests over executed ones
    - $\theta_i = p_i \varphi_i$: weighted failure rate = *unreliability* contribution
  - θ_i^k *at each iteration k* is used to direct more testing to partitions that are expected to contribute more to improve *delivered reliability*
  - θ_i^k can be subject to strong variation **=>Adaptive sampling**

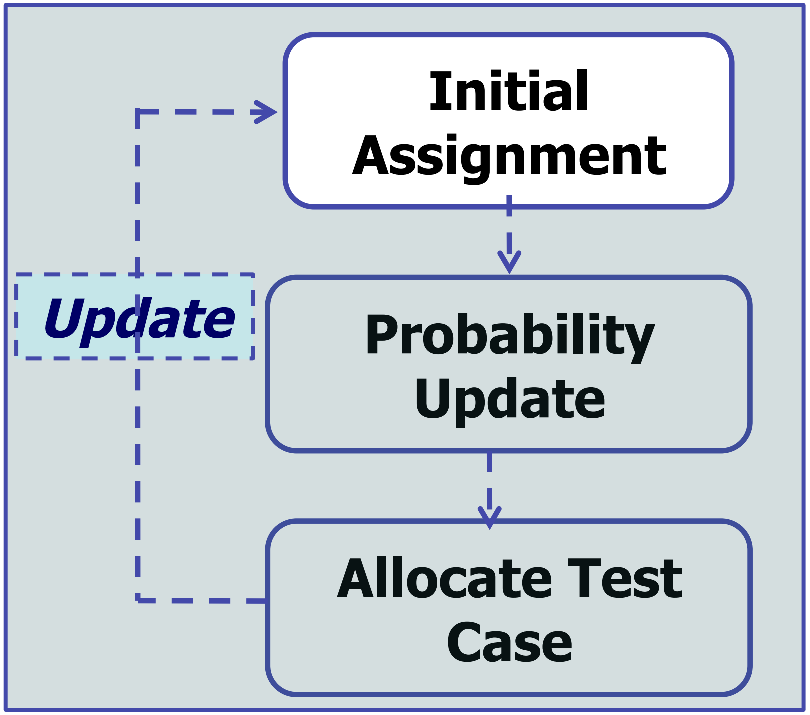
Importance Sampling

Test Allocation

- **Importance Sampling method**
  - Progressively approximate true (unknown) distribution of a variable
  - Beliefs (i.e., hypotheses) about the distribution represented by “samples”
  - At each iteration, more samples (i.e., tests) drawn from the best “hypothesis”
  - *“Best” in covrel=> Distribution to maximize exp. reliability contribution*
- ***Steps***
  1. **Initial allocation**
  2. **Probability Update**
  3. **Assignment**

Importance Sampling

Test Allocation



1 ***Initial allocation***

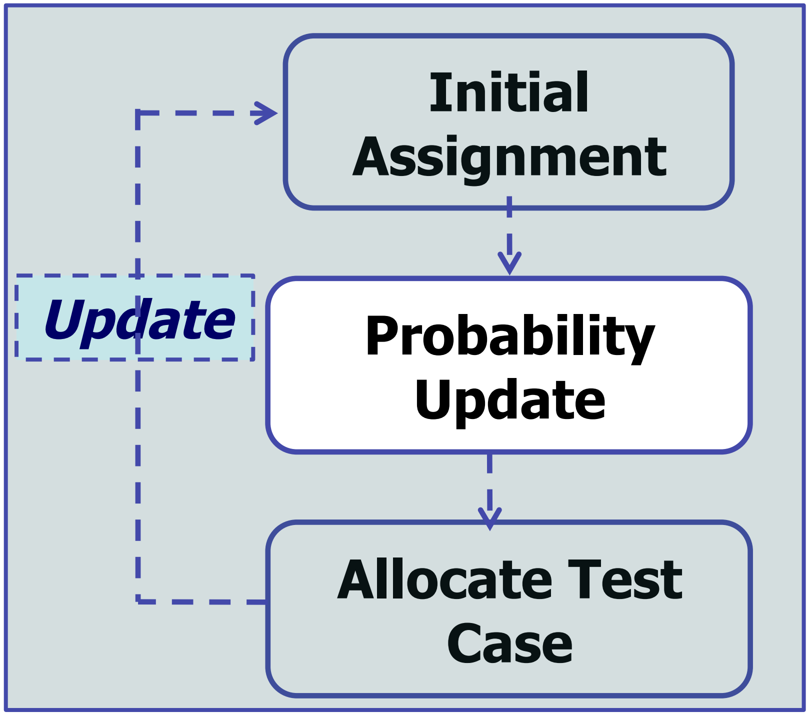
- Assuming no domain-specific knowledge => **proportional-to-usage allocation**

$$T_i^{(0)} \approx T^{(0)} \frac{p_i}{\sum_{i=1}^{m} p_i}$$

- $T^{(0)}$ small percentage of T to trigger the algo (e.g., 5%)

Importance Sampling

Test Allocation



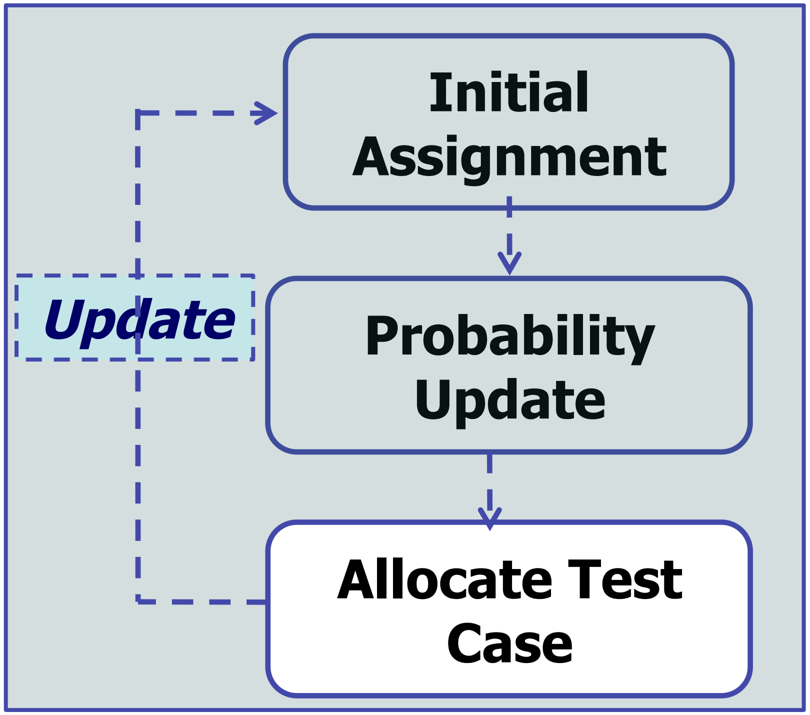
2 **Probability update**

$$\pi_i^{(k)} = \gamma \pi_i^{(k-1)} + (1-\gamma)(1-\theta_i^{(k-1)})$$

- $\boldsymbol{\pi_i^{(k)}}$ => *relative importance given to D_i* at iteration *k*
- $\boldsymbol{\gamma}$ => *factor regulating the importance of the past w.r.t. to current observations*

Importance Sampling

Test Allocation



3 *Assignment*

$$T_i^{(k+1)} \approx T^{(k+1)} \pi_i^{(k)}$$

- *$T^{(k+1)}$ computed by **KLD Adaptation*** [1] *as $T^{(k+1)} = f(Error, Confidence)$*
- *Distribute $T^{(k+1)}$ by sampling according to $\boldsymbol{\pi_i^{(k)}}$*
- ***Output**: $T_i^{(k+1)}$*

[1] D. Fox. Adapting the Sample Size in Particle Filters Through KLD-Sampling. Int. Journal of Robotics Research, 22:2003, 2003.

Within-partition coverage-based Test Selection

Test Selection

Initial Selection

- Random selection

Compute spectrum

- *covrel* computes the cumulative count spectrum achieved so far

Order entities

- Ordered by their frequency of usage
  - 1/3 of entities to *high* frequency group
  - 1/3 to *medium*
  - 1/3 to *low*

Rank entities

- Weights assigned to each group
  - W(high)= 10^{-1}
  - W(medium) = 10^{0}
  - W(low) = 10^{1}

Select tests

- Test cases with the highest ranks are selected

Next Iteration

3 Coverage Criteria:

Function, Statement, Branch

COVREL

EVALUATION

Empirical Evaluation. Setting (1/2)

Research question

- *Is covrel more effective at reliability improvement than traditional operational profile-based testing?*

Factors

- **Subjects**
  - 4 programs - 18 versions from SIR
- **Test suite** available from SIR
- **Partitions** = Functionalities
- **Fault matrices** generations: “easy” and “hard” faults matrix
- **Operational Profile**. 50 randomly generated profiles
- **Coverage criteria**: function, branch, and statement

Empirical Evaluation. Setting (2/2)

- **Number of runs**: 50 profiles x 2 techniques x 18 subjects x 2 matrices x 3 criteria = 10800

- **Evaluation metrics** (*covrel vs* operational testing)
  1. Number of test cases required to reach the maximum reliability achievable with the test suite
  2. Values of reliability achieved with equal number of test cases

- Further evaluation: *covrel vs coverage-based testing*

Empirical Evaluation. Subjects

Program	LoC	Vers.	Tests cases	Seeded Faults	Detectable Faults	"Hard" Faults
Grep	9463	v1	809	18	5	4
Grep	9987	v2	809	8	4	4
Grep	10124	v3	809	18	8	5
Grep	10143	v4	809	12	3	3
Gzip	4594	v1	214	16	7	6
Gzip	5083	v2	214	7	3	1
Gzip	5233	v4	214	12	3	3
Gzip	5745	v5	214	14	5	4
Sed	9867	v2	360	5	5	3
Sed	7146	v3	360	6	6	5
Sed	7086	v4	363	4	1	1
Sed	13398	v5	370	4	4	4
Sed	13413	v6	370	6	6	6
Sed	14456	v7	370	4	4	4
Flex	9558	v1	567	19	16	8
Flex	10274	v2	670	20	13	9
Flex	10296	v3	670	17	9	9
Flex	11447	v4	670	16	11	8

Results: TCs to reach “reliability=1”

#of wins, losses and ties of *covrel* vs OT over **50 runs**

Configuration	Outcome	Mean	Median
Conf. 1: Function-Hard FM	**Wins**	**35.5**	**40**
	Losses	**12.92**	**8.5**
	Ties	**1.57**	**0**
Conf. 2: Branch-Hard FM	**Wins**	**35.14**	**39**
	Losses	**12.78**	**10**
	Ties	**2.07**	**0**
Conf. 3: Statement-Hard FM	**Wins**	**36.14**	**41**
	Losses	**11.85**	**8.5**
	Ties	**2**	**0**
Conf. 4: Function-Default FM	**Wins**	**34.16**	**37**
	Losses	**14.16**	**9**
	Ties	**1.66**	**0**
Conf. 5: Branch-Default FM	**Wins**	**33.05**	**36.5**
	Losses	**15.88**	**13**
	Ties	**1.05**	**0**
Conf. 6: Statement-Default FM	**Wins**	**34.38**	**39.5**
	Losses	**14.38**	**9.5**
	Ties	**1.22**	**0.5**

- In the average, *Covrel* > OT in all configurations
- *Covrel* > OT in 77 out of 96 scenarios
- Summing over repetitions, *Covrel* > OT in all conf.
- Summing over repetitions, Covrel > OT for 14 out of 18 subjects

Statistical comparison

Pairwise Comparison		
	covrel	**OT**
Mean	**34.62**	**13.81**
Median	**39.00**	**9.50**
P-value	**3.1270e-09**	-

Results: Reliability growth

- Reliability growth at three checkpoints: 10%-50%-90% of test cases needed to achieve reliability 1

Number of wins in terms of greater reliability at 10/50/90% of tests			
	10%	***50%***	***90%***
Covrel Mean	11.40	17.12	16.26
Covrel Median	8.5	11.85	13.77
OT Mean	7	16	15
OT Median	8	9	10
Covrel-OT P-value	*0.2168*	***0.0087***	*0.4156*

- In the average, covrel>OT for all configurations
- Expectation: *covrel* > OT for higher values of reliability
  - $(\mathit{Covrel} - \mathrm{OT})_{50\%} > (\mathit{Covrel} - \mathrm{OT})_{90\%} > (\mathit{Covrel} - \mathrm{OT})_{10\%}$
  - Most of improvement of *covrel* is **between 90% and 100%**
  - $(\mathit{Covrel} - \mathrm{OT})_{hardFaults} > (\mathit{Covrel} - \mathrm{OT})_{AllFaults}$

covrel vs coverage-based testing comparison

- N° tests to get maximum attainable reliability
- '-' means maximum attainable coverage achieved before removing all faults
- *Greedy total* and *greedy additional*

- ***Covrel > total* in 85% of cases**
- ***Covrel > additional* in 50% of cases**

Program	Function		Branch		Statement	
	covrel	*total*	*covrel*	*total*	*covrel*	*total*
Grep v1	**304**	547	**130**	-	**113**	574
Grep v2	**623**	-	**321**	531	**358**	501
Grep v3	**72**	-	**156**	487	**97**	477
Grep v4	**611**	728	**108**	724	**108**	728
Gzip v1	**178**	205	**118**	205	**124**	206
Gzip v2	26	3	28	1	29	3
Gzip v4	**25**	208	**26**	207	**25**	207
Gzip v5	**62**	204	**66**	207	**60**	206
Sed v2	**84**	-	**114**	92	**72**	88
Sed v3	**62**	-	**44**	352	**42**	352
Sed v4	**47**	-	**26**	123	**45**	131
Sed v5	**11**	-	**11**	362	**10**	362
Sed v6	**70**	-	**47**	104	**60**	89
Sed v7	**50**	-	**42**	-	**44**	-
Flex v1	**18**	411	**16**	379	**18**	382
Flex v2	**276**	669	**354**	669	**314**	664
Flex v3	**542**	-	**611**	614	622	618
Flex v4	96	4	228	4	257	4

Threats

- **Internal**
  - Profile representativeness
  - Subjects' test suites limitations
- **Construct**
  - Measured testing reliability ≠ operational reliability
- **External**
  - Representativeness of subjects and faults
  - Programs with similar features (e.g., same language, small size)
  - Different versions are not different programs

Verifiability

- **Artifacts for automating experiments**
  - Prototype written in Java and Python
  - http://labsedc.isti.cnr.it/covrel2017
  - Takes the subject program and version, and the number of repetitions (i.e., of profiles to generate)
  - Detailed results in CSV files, one per configuration
  - Operating instructions to test other SIR programs

Relation to GAUSS and future work

- **Challenges and next steps**
  - **Test case generation**
    - Spectra collection/online learning to generate tests
  - *Adaptive allocation* and *"white-box" test selection* are decoupled – further approaches can be experimented to improve both tasks
  - Application to **larger applications**
    - Apps possibly closer to GAUSS target
    - Scalability
  - **Runtime testing**
  - Iterative closed-loop approach enables runtime testing
    - Adaptiveness w.r.t. runtime data rather than (or in addition to) test data
    - Runtime data would enable a better approximation of profiles
    - Reliability *assessment*, besides improvement

Thanks for the attention !
Questions ?