SEARCH-BASED TEST OPTIMIZATION FOR SOFTWARE SYSTEMS

Shaukat Ali
Senior Research Scientist
Simula Research Laboratory, Norway
GECCO 2018, Kyoto, Japan

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07.
https://doi.org/10.1145/3205651.3207883

OUTLINE OF THE TUTORIAL

- Theoretical Foundation
  - ✓ Multi-Objective Search Problems
  - ✓ Multi-Objective Test Optimization
- Applications in Real-World Applications
  - ✓ Test Optimization for Cyber-Physical Systems
  - ✓ Test Minimization in Product Lines
  - ✓ Test Prioritization in Product Lines
  - ✓ Test Case Selection in the Maritime Domain
  - ✓ Test Prioritization in Communication Domain
  - ✓ Test Optimization with Search and Machine Learning
- Tools/Frameworks

Theoretical Foundations

Content from: S. Ali, H. Lu, S. Wang, T. Yue and M. Zhang, Uncertainty-wise Testing of Cyber-Physical Systems, Advances in Computers, Vol 107, Chapter 2, 2017

MULTI-OBJECTIVE SEARCH PROBLEM

- $Obj = \{o_1, o_2, .., o_{non}\}$
- Each o_i is measured with a Cost, Effectiveness, and Efficiency measure
- To decrease the cost, increase the effectiveness, and/or increase the efficiency.

MEASURES

- *CostMeasure* = {cm_1, cm_2, …, cm_{ncm}}
- *EffectMeasure* = {em_1, em_2, …em_{nem}}
- *EfficiencyMeasure* = {ec_1, ec_2, …, ec_{nec}}
  - ✓Efficiency measure: *Effectiveness* per *Cost*.
  - ✓If all combinations are valid then *nec* = *ncm***nem*.
  - ✓Else: *nec* < *ncm***nem*.

[simula . research laboratory]

SEARCH PROBLEM (1/2)

- *PS* = {ps_1, ps_2, …, ps_{nps}}.
- Suppose, we have three functions:
  - ✓*CMF*(ps_i)
  - ✓*EMF*(ps_i)
  - ✓*ECF*(ps_i)

[simula . research laboratory]

SEARCH PROBLEM (2/2)

- Finding the best solution ps_k out of *PS* :
  - ✓*Condition 1:* $\forall_{cm}$ in *CostMeasure*, $\forall_{ps}$ in *PS* - ps_k, *CMF*(ps_k, *cm*) <= *CMF*(*ps*, *cm*).
  - ✓*Condition 2:* $\forall_{em}$ in *EffectMeasure*, $\forall_{ps}$ in *PS* - ps_k, *EMF*(ps_k, *em*) >= *EMF*(*ps*, *em*).
  - ✓*Condition 3:* $\forall_{ec}$ in *EfficiencyMeasure*, $\forall_{ps}$ in *PS* - ps_k, *ECF*(ps_k, *ec*) >= *ECF*(*ps*, *ec*).

[simula . research laboratory]

TEST OPTIMIZATION

- A multi-objective test optimization optimizes a set of test cases *TC* = {tc_1, tc_2, …, tc_{ntc}},
- To optimize based on cost, effectiveness, and efficiency.

[simula . research laboratory]

TEST OPTIMIZATION: TEST CASE ATTRIBUTES (1/2)

- *Cost*, *Effectiveness*, *Efficiency*, and *Uncertainty* attributes :
  - ✓ $CostAtt = \{cat_1, cat_2, .., cat_{ncat}\}$, where each cat_i has exactly one type such as *Integer*, *Real*, and *Boolean* or an advanced data type.
  - ✓ $EffectAtt = \{efat_1, efat_2, .., efat_{nefat}\}$
  - ✓ $EfficiencyAtt = \{ecat_1, ecat_2, .., ecat_{necat}\}$

TEST OPTIMIZATION: TEST CASE ATTRIBUTE VALUES (2/2)

- Each test case tc_i in *TC* may have three sets of associated values
  - ✓ $CostVal_i = \{cvali_1, cvali_2, \ldots, cvali_{ncat}\}$
  - ✓ $EffectVal_i = \{efvali_1, efvali_2, \ldots, efvali_{nefat}\}$
  - ✓ $EfficiencyVal_i = \{ecvali_1, ecvali_2, \ldots, ecvali_{necat}\}$

TEST OPTIMIZATION PROBLEMS

- A solution ps_k in *PS*, is a set of test cases from *TC*,
- A subset: Test Minimization Problem
- The same set of test cases with best order to execute: Test Prioritization problem

CALCULATION OF OBJECTIVES

- Simple way: take an average of the solution. For example, a cost measure cm_i corresponding to a cost attribute, e.g., cat_k can be calculated as follows:
  - ✓ $cm_i = (\sum_{j=1 \text{ to } ntc} cvalj_k)/ntc$, where $cvalj_k$ represents a value for a j_{th} test case in *TC* corresponding to the cat_k cost attribute.
- Similarly for other types of objectives

NORMALIZATION

- Each objective, i.e., cm_i, em_i, ec_i, and um_i, may take a value in different ranges.
- Two commonly used normalization functions:
  - ✓*Condition 1*: If maximum and minimum values for a calculated objective are known, then:
    - $N1(x) = (x - x_{min}) / (x_{max} - x_{min})$
  - ✓*Else*
    - $N2(x) = (x/(x+ \beta))$, where β is any value greater than 0

TEST OPTIMIZATION PROBLEM

- Minimize cost and maximize effectiveness and efficiency,
- $\forall_{cm}$ in *CostMeasure*, Minimize(*cm*) && $\forall_{em}$ in *EffectMeasure*, Maximize(*em*) && $\forall_{ec}$ in *EfficiencyMeasure*, Maximize(ec).

TEST MINIMIZATION PROBLEM

- Find a minimum number of test cases (*mtc*) out of the total test cases *ntc* in *TC*.
- *mtc* in a solution can be any combination of a number of test cases in *TC*, i.e., from *1* to *ntc-1*.
- Possible minimization solutions :
  - ✓$nps = {}^{ntc}C_1 + {}^{ntc}C_2 + .. + {}^{ntc}C_{ntc-1} = 2^{ntc} - 1$
  - ✓$PS = \{ps_1, ps_2, \ldots, ps_{2^{ntc}-1}\}$

TEST MINIMIZATION PROBLEM

- If *mtc*=*ntc*, no minimization at all.
- A cost objective must be defined
  - ✓TMP: calculate the percentage of test set minimization, TMP = 1 - ntc_i/ntc,
  - ✓NMTC = ntc_i is the number of minimized test cases in a solution ps_i.
    - May need a normalization function
- Search will favor 0 test cases
  - ✓Solution: Set a minimum number of test cases

TEST MINIMIZATION PROBLEM

- Finding the best solution ps_k from PS which has a minimum number of test cases that meet :
  - ✓*Condition 1:* $\forall_{cm}$ in *CostMeasure*, $\forall_{ps}$ in $PS - ps_k$, $CMF(ps_k, cm)$ <= $CMF(ps, cm)$ and $mps_k < mps$ and $mps_k \neq 0$
  - ✓*Condition 2:* $\forall_{em}$ in *EffectMeasure*, $\forall_{ps}$ in $PS - ps_k$, $EMF(ps_k, em)$ >= $EMF(ps, em)$ and $mps_k < mps$ and $mps_k \neq 0$.
  - ✓*Condition 3:* $\forall_{em}$ in *EfficiencyMeasure*, $\forall_{ps}$ in $PS - ps_k$, $ECF(ps_k, ec)$ >= $ECF(ps, ec)$ and $mps_k < mps$ and $mps_k \neq 0$.

TEST PRIORITIZATION

- Prioritizing the test cases in a specific order to execute.
- Test set minimization and test case prioritization may be combined together.
  - ✓Minimize test cases and then prioritize the minimized test cases
  - ✓Prioritize test cases and then minimize them
  - ✓Prioritize and minimize the test cases at the same time.

TEST PRIORITIZATION

- Find a best order to execute test cases that meet all the objectives.
- Prioritization keeps the number of test cases to the same and thus the possible prioritization solutions :
  - ✓$nps = (ntc)*(ntc-1)*(ntc-2)*...1 = ntc!$
  - ✓$PS = \{ps_1, ps_2, ..., ps_{ntc!}\}$

TEST PRIORITIZATION

- A specific objective for prioritization must be defined:
- Prioritization Impact (*PI*)
  - ✓$PI_i = (ntc-p+1)/ntc$,
- Position
  - ✓p, where p ranges from 1 to ntc
  - ✓Normalization is required

TEST PRIORITIZATION

- Time budget
  - ✓E.g., a certain percentage of test cases.
  - ✓Based on the actual execution time, such as in maximum number of hours.

TEST PRIORITIZATION

- Prioritization problem: Finding the best solution ps_k :
  - ✓*Condition 1:* $\forall_{cm}$ in *CostMeasure*, $\forall_{ps}$ in *PS* - ps_k, *CMF*(ps_k, *cm*) <= *CMF*(*ps*, *cm*) and *tbg* <= *bg*,
  - ✓*Condition 2:* $\forall_{em}$ in *EffectMeasure*, $\forall_{ps}$ in *PS* - ps_k, *EMF*(ps_k, *em*) >= *EMF*(*ps*, *em*) and *tbg* <= *bg*
  - ✓*Condition 3:* $\forall_{em}$ in *EfficiencyMeasure*, $\forall_{ps}$ in *PS* - ps_k, *ECF*(ps_k, *ec*) >= *ECF*(*ps*, *ec*)

Test Optimization for Cyber-Physical Systems

- Content from: S. Ali, H. Lu, S. Wang, T. Yue and M. Zhang, Uncertainty-wise Testing of Cyber-Physical Systems, Advances in Computers, Vol 107, Chapter 2, 2017
- Content from: S. Ali, Y. Li, T. Yue and M. Zhang, An Empirical Evaluation of Mutation and Crossover Operators for Multi-Objective Uncertainty-Wise Test Minimization, 10th International Workshop on Search-based Software Testing, 2017
- Content from: M. Zhang, Y. Li, S. Ali and T. Yue, Uncertainty-Wise and Time-Aware Test Case Prioritization with Multi-Objective Search, Simula Research Laboratory, Technical Report (2017-03), 2017

CYBER-PHYSICAL SYSTEMS (CPSS) ARE ALL AROUND US

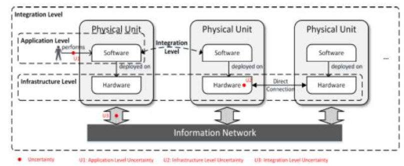
- Applications, e.g., Healthcare, Aerospace, Avionics, Oil/gas and Maritime, Industrial Automation, and Tele-communication
- Daily life CPS applications forecasted > $82 trillion by 2025 (alone in USA)
- CPSs must be dependable
- Improving CPSs' dependability with automated testing

Evans, P.C., Annunziata, M.: Pushing the Boundaries of Minds and Machines. General Electric (GE), (2012)

UNCERTAINTY IS INHERENT IN CPSS.

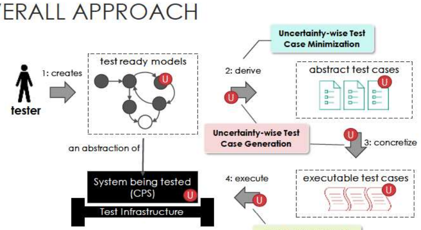
- Unpredictable physical environment
- CPSs are typically black-boxes with/without direct access via APIs
- Interactions among software, hardware, agents (human, animals, ..)
- Expected behavior of a CPS is uncertain in the face of uncertainty in environment

UNCERTAINTY IN **CPS** CAN OCCUR AT THREE LOGICAL LEVELS: APPLICATION, INFRASTRUCTURE AND INTEGRATION

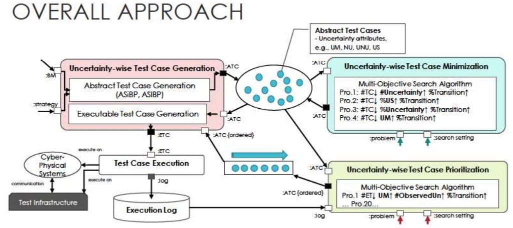


Content from: Zhang M, Ali S, Yue T, Norgren R and Okariz O (2017), "Uncertainty-Wise Cyber-Physical System Test Modeling", Software & Systems Modeling (SoSyM), DOI: https://doi.org/10.1007/s10270-017-0609-6, Springer

OVERALL APPROACH



OVERALL APPROACH



BM is test ready model, ATC is abstract test case, ETC is executable test case

CONCEPTS RELATED TO TEST CASE GENERATION

- Uncertainty Theory is "*a branch of mathematics for modeling human uncertainty*" [1].
- *Uncertainty Measure (UM)* [1] is defined in *Uncertainty Theory* that is a specific value (a number) assigned to the event for indicating the *belief degree* of the occurrence of the event.

Axiom 1. (Normality) $\mathcal{M}(\Gamma) = 1$, (Γ is the universal set)

Axiom 2. (Duality) $\mathcal{M}\{\Lambda\} + \mathcal{M}\{\Lambda^c\} = 1$, where Λ shows a particular event, whereas Λ^c shows all the elements in the universal set excluding Λ.

Axiom 3. (Subadditivity) $\mathcal{M}\{\bigcup_{i=1}^{\infty} \Lambda_i\} < \sum_{i=1}^{\infty} \mathcal{M}\{\Lambda_i\}$ (every *countable* sequence of events $\Lambda_1, \Lambda_2, \ldots$)

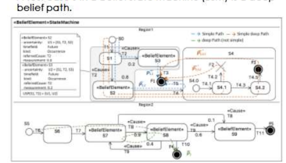
Uncertainty Space: A triplet $(\Gamma, \mathcal{L}, \mathcal{M})$, where Γ is the universal set, $\mathcal{L}$ is a σ-algebra [16] over Γ, and $\mathcal{M}$ is *UM*.

Theorem: Let $(\Gamma_k, \mathcal{L}_k, \mathcal{M}_k)$ be uncertainty spaces and $\Lambda_k \in \mathcal{L}_k$, for $k = 1, 2, \ldots n$. Then $\Lambda_1, \Lambda_2, \ldots \Lambda_n$ are always independent of each other if they are from different uncertainty spaces.

[1]. B. Liu, Uncertainty theory, Springer, 2015.

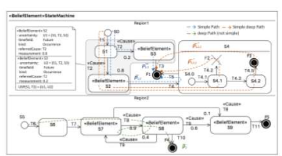
TWO TEST GENERATION STRATEGIES ARE DEFINED

A test case *t* in a belief state machine (BSM) is a *deep* belief path.



- ***All Simple Belief Path Coverage (ASiBP).*** Test set *T* satisfies *ASiBP* on BSM if and only if any belief simple deep path $\bar{P}'$ from *initial state* to one of *final states* in BSM is in *T*.
- ***All Specified Length Belief Path Coverage (ASIBP).*** Test set *T* satisfies *ASIBP* on belief state machine (BSM) if and only if any belief simple deep path $\bar{P}'$ of length less than *specified length* from *initial state* to one of *final states* in BSM is in *T*.
  - *The uncertainty (S8, T9, S7) is only possible to be covered by* ***ASIBP***.

EACH GENERATED TEST CASE HAS UNCERTAINTY-RELATED ATTRIBUTES.



A Test Case

t1 = (*S0, T1, S1, T2, S3, T3, F1*)

- *number of transitions covered in t1*
  - *3, i.e., TR(t1)={T1, T2, T3}*

Uncertainty-related attributes

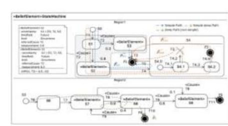
- *number of uncertainties covered in t1 (NU)*
  - *U2={S1, T2, S3}, and NU(t1) = 1*
- *number of unique uncertainties covered in t1 (NUU)*
  - *U2={S1, T2, S3}, and NUU(t1) = 1*
- *number of uncertainty space covered in t1 (NUSP)*
  - *USP(S1, T2), and NUSP(t1) =1*
- *uncertainty measure (UM) of t1*
  - *0.2*

$UM(t1) = \mathcal{M}\{S0,T1,S1\} \wedge \mathcal{M}\{S1,T2,S3\} \wedge \mathcal{M}\{S3,T3,F1\} = 1 \wedge 0.2 \wedge 1 = 0.2$

After executing t1,

- *number of uncertainties observed of t1 (NOU)*
  - *If NOU(t1) = 1, it means that one uncertainty is observed, i.e., S1 transited to S3 after T2 was executed*
- *time for executing t1 (ET)*

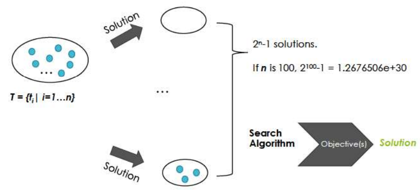
EXAMPLE OF ABSTRACT TEST CASE GENERATION USING *ASIBP*



#	Abstract Test Case
1	R1:(S0, T1, S1, T2, S3, T3, F1) R2:(S5, T6, S6, T7, S7, T8.1, S8, T9, S9, T11, F5)
2	R1:(S0, T1, S1, T2, S3, T3, F1) R2:(S5, T6, S6, T7, S7, T8, S8, T10, F4)
3	R1:(S0, T1, S1, T2, S3, T3, F1) R2:(S5, T6, S6, T7, S7, T8, S9, T11, F5)
4	R1:(S0, T1, S1, T2, S2, T4, S4.0, T4.1, S4.1, T4.3, S4.2, T4.5, F3, T5, F1) R2:(S5, T6, S6, T7, S7, T8, S8, T9, S9, T11, F5)
5	R1:(S0, T1, S1, T2, S2, T4, S4.0, T4.1, S4.1, T4.3, S4.2, T4.5, F3, T5, F1) R2:(S5, T6, S6, T7, S7, T8, S8, T10, F4)
6	R1:(S0, T1, S1, T2, S2, T4, S4.0, T4.1, S4.1, T4.3, S4.2, T4.5, F3, T5, F1) R2:(S5, T6, S6, T7, S7, T8, S9, T11, F5)
7	R1:(S0, T1, S1, T2, S2, T4, S4.0, T4.1, S4.1, T4.2, F2) R2:(S5, T6, S6, T7, S7, T8, S8, T9, S9, T11, F5)
8	R1:(S0, T1, S1, T2, S2, T4, S4.0, T4.1, S4.1, T4.2, F2) R2:(S5, T6, S6, T7, S7, T8, S8, T10, F4)
9	R1:(S0, T1, S1, T2, S2, T4, S4.0, T4.1, S4.1, T4.2, F2) R2:(S5, T6, S6, T7, S7, T8, S9, T11, F5)

Note that the uncertainty, (*S8, T9, S7*), is impossible to be covered by *ASiBP*

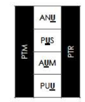
UNCERTAINTY-WISE TEST CASE MINIMIZATION



[simula . research laboratory] 33

FOUR UNCERTAINTY-WISE TEST CASE MINIMIZATION PROBLEMS WERE PROPOSED BASED ON SIX OBJECTIVES.

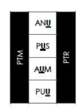
- U1: Average Normalized Number of Uncertainties Covered (**ANU**): $\frac{\sum_{i=1}^{mt} nor(NU(t_i))}{mt}$
  where $nor(x) = \frac{x}{x+1}$, *mt* is the number of test cases in the minimized test set.
- U2: Percentage of Uncertainty Space Covered (**PUS**): $\frac{musp}{nusp} \times 100\%$
  Where *musp* is the number of uncertainty spaces covered in the minimized test set and *nusp* is the number of uncertainty spaces covered in the test ready model.
- U3: Average Overall Uncertainty Measure (**AUM**): $\frac{\sum_{i=1}^{mt} UM(ti)}{mt} \times 100\%$
  Where *mt* is the number of test cases in the minimized test set



[simula . research laboratory] 34

FOUR UNCERTAINTY-WISE TEST CASE MINIMIZATION PROBLEMS WERE PROPOSED BASED ON SIX OBJECTIVES.

- U4: Percentage of Unique Uncertainties Covered (**PUU**): $\frac{muu}{nuu} \times 100\%$
  Where *muu* is the number of unique uncertainties covered in the test set and *nuu* is the number of unique uncertainties covered in the test ready model.
- C1: Percentage of Test Case Minimization (**PTM**) : $\frac{nt}{mt}$
  where *nt* is the number of test cases in the generated test set, and *mt* is the number of test cases in the minimized test set.
- E1: Percentage of Transition Coverage (**PTR**): $\frac{mtr}{ntr} \times 100\%$
  Where *mtr* is the number of transitions covered in the minimized test set and *ntr* is the number of transitions covered in the test ready model.



[simula . research laboratory] 35

- 9 test cases (***nt*** = 9), denoted as ***T = {t_i | i=1…9}***, are generated by ***ASiBP*** based on the belief state machine, **BSM**.
  - The BSM contains 6 uncertainties (***nuu*** = 6), denoted as ***U= {u_j | j=1…6}***, 3 uncertainty spaces (***musp*** = 3), denoted as ***USP = {usp_k | k=1…3}***, and 18 transitions (***ntr*** = 18), denoted as ***TR = {tr_l | l =1…18}***.
  - Each generated test case contains the following information: $Us(t_i)$, $UM(t_i)$, $USP(t_i)$, $TR(t_i)$.

Assume that a solution is ***MT*** = {t_1, t_7}

#	Abstract Test Case	Us	UM	USP	TR
1	R1:(S0, T1, ***S1***, ***T2***, ***S3***, T3, F1) R2:(S5, T6, S6, T7, S7, T8.1, ***S8***, ***T9***, ***S9***, T11, F5)	u_2, u_6	0.2	usp_1, usp_3	tr_1, tr_2, tr_3, tr_6, tr_7, tr_{12}, tr_9, tr_{11}
7	R1:(S0, T1, ***S1***, ***T2***, ***S2***, T4, S4.0, T4.1, S4.1, T4.2, F2) R2:(S5, T6, S6, T7, ***S7***, ***T8***, ***S8***, ***T9***, ***S9***, T11, F5)	u_1, u_3, u_6	0.2	usp_1, usp_2, usp_3	tr_1, tr_2, tr_4, tr_{13}, tr_{14}, tr_6, tr_7, tr_8, tr_9, tr_{11}

- ***mt*** = 2
- $NU(t_1) = 2$, $NU(t_7) = 3$, nor $(NU(t_1)) = \frac{2}{2+1}$ and $nor(NU(t_7)) = \frac{3}{3+1}$
- $Us(t_1)$ U $Us(t_7)$ = {u_1, u_2, u_3, u_6}, ***muu*** = *4*
- $USP(t_1)$ U $USP(t_7)$ = {usp_1, usp_2, usp_3}, ***musp*** = *3*
- $TR(t_1)$ U $TR(t_7)$ = {tr_1, tr_2, tr_3, tr_4, tr_6, tr_7, tr_8, tr_9, tr_{11}, tr_{12}, tr_{13}, tr_{14}}, ***mtr***=*12*

Prob.1: ANU $= \frac{\frac{2}{2+1}+\frac{3}{3+1}}{2}$

Prob.2: PUS $= \frac{3}{3}$

Prob.3: AUM $= \frac{0.2+0.2}{2}$

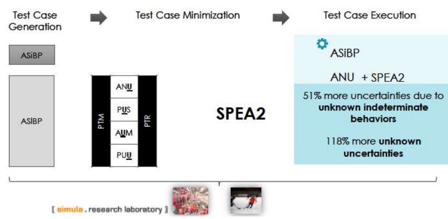
Prob.4: PUU $= \frac{4}{6}$

PTM $= \frac{2}{9}$

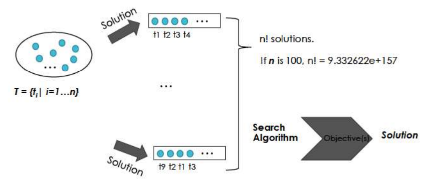
PTR $= \frac{12}{18}$

[simula . research laboratory] 36

BASED ON THE CURRENT EVALUATION, **AMP+ANU+SPEA2** IS THE BEST STRATEGY IN ORDER TO DISCOVER UNCERTAINTIES.

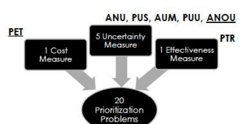


UNCERTAINTY-WISE TEST CASE PRIORITIZATION



OBJECTIVES

- To guide the search to prioritize test cases with high cost-effectiveness in earlier positions to be executed
  - *PI (i)* measures the impact of position *i* of test case *t*: $\frac{mt-i+1}{mt}$, where *mt* is the number of test cases formed as a sequence, and *i* is the position of t in this sequence.



- **Percentage of Execution Time (PET)** : $\frac{\sum_i^{mt} ET(t_i) \times PI(i)}{ET_{total}}$, where ET_{total} is the total of the execution time for the whole set.
- **Average Number of Observed Uncertainties (ANOU)**: $\frac{\sum_i^{mt} NOU(t_i) \times PI(i)}{mt}$

- Assume that the time of executing 9 test cases (***nt*** = 9) is 320s
  - ❑ Each executed test case contains the following information: ET(t_i), NOU(t_i)

#	Abstract Test Case	Us	UM	USP	TR	ET	NOU	Solution Position	PI
1	R1:{S0, T1, S1, T2, S3, T3, F1} R2:{S5, T6, S6, T7, S7, T8.1, S8, T9, S9, T11, F5}	u_2, u_6	0.2	usp_1, usp_3	tr_1, tr_2, tr_3, tr_6, tr_7, tr_{12}, tr_9, tr_{11}	20s	1	2	$\frac{9-2+1}{9}$
…									
7	R1:{S0, T1, ***S1***, ***T2***, ***S2***, T4, S4.0, T4.1, S4.1, T4.2, F2} R2:{S5, T6, S6, T7, ***S7***, ***T8***, ***S8***, ***T9***, ***S9***, T11, F5}	u_1, u_3, u_6	0.2	usp_1, usp_2, usp_3	tr_1, tr_2, tr_4, tr_{13}, tr_{14}, tr_6, tr_7, tr_8, tr_9, tr_{11}	35s	2	1	$\frac{9-1+1}{9}$
…									

Solution= (t_7, t_1, | t_5, t_4, t_3, t_2, t_9, t_6, t_8)

Time budget, i.e., 20%

$\Delta TR(t_1') = TR(t_7) = \{tr_1, tr_2, tr_4, tr_{13}, tr_{14}, tr_6, tr_7, tr_8, tr_9, tr_{11}\}$, $\Delta ntr_1' = 10$

$\Delta TR(t_2') = (TR(t_7) \cup TR(t_1)) / TR(t_7) = \{tr_3, tr_{12}\}$, $\Delta ntr_2' = 2$

$\Delta Us(t_1') = Us(t_7) = \{u_1, u_3, u_6\}$, $\Delta nuu_1' = 3$

$\Delta Us(t_2') = (Us(t_7) \cup Us(t_1)) / Us(t_7) = \{u_2\}$, $\Delta nuu_2' = 1$

$$PET = \frac{35*1+20*\frac{8}{9}+\cdots+ET(t_9)*\frac{1}{9}}{320} \qquad PTR = \frac{10*1+2*\frac{8}{9}+\cdots+\Delta ntr_9'*\frac{1}{9}}{18}$$

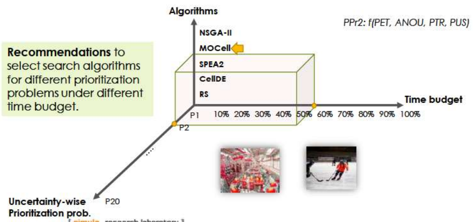
$$ANU = \frac{\frac{2}{2+1}*1+\frac{1}{1+1}*\frac{8}{9}+\cdots+nor(NU(t_9))*\frac{1}{9}}{9} \qquad PUS = \frac{\frac{3}{3}*1+\frac{2}{3}*\frac{8}{9}+\cdots+UM(t_9)*\frac{1}{9}}{9}$$

$$AUM = \frac{0.2*1+0.2*\frac{8}{9}+\cdots+UM(t_9)*\frac{1}{9}}{9} \qquad PUU = \frac{3*1+1*\frac{8}{9}+\cdots+\Delta nuu_9'*\frac{1}{9}}{6}$$

$$ANOU = \frac{\frac{2}{2+1}*1+\frac{1}{1+1}*\frac{8}{9}+\cdots+nor(NOU(t_9))*\frac{1}{9}}{9}$$

FOUR MULTI-OBJECTIVE SEARCH ALGORITHMS UNDER VARIOUS TIME BUDGETS WERE COMPARED

Recommendations to select search algorithms for different prioritization problems under different time budget.



Test Minimization in Product Lines

Content from: S. Wang, S. Ali, A. Gotlieb. Cost-Effective Test Suite Minimization in Product Lines Using Search Techniques. Journal of Systems and Software (JSS) vol. 103, pp. 370-391, 2015

PROBLEM AND CASE STUDIES

- Multi-Objective Test Suite Minimization Problem
  - ✓Eliminate the redundancy of a test suite
  - ✓Several concerns when minimization
    - Feature pairwise coverage, fault detection capability
- Case Studies
  - ✓Industrial: Cisco with four products
  - ✓500 artificial problems

OBJECTIVES

- Cost Measure
  - ✓ **OET**: Overall execution time taken by a minimized set of test cases
- Three Effectiveness Measures
  - ✓ **TMP**: Test minimization percentage to measure the amount of reduction for the number of test cases
  - ✓ **FPC**: Feature pairwise coverage to measure how much pairwise coverage can be achieved by a minimized set of test cases
  - ✓ **FDC**: Fault detection capability
    - ✓ Rate of successful executions in a given time (e.g., a week)
  - ✓ **AEF**: Average execution frequency based on the execution frequency of each included test cases

SEARCH ALGORITHMS USED

Different mechanisms			Algorithms
Evolutionary Algorithms (EAs)	GAs	Weight-Based GA	WBGA
			WBGA-MO
			RWGA
		Sorting-Based GA	NSGA-II
		Cellular-Based GA	MOCell
	Strength Pareto EA		SPEA2
	Evolution Strategies		PAES
Swarm Algorithm	Particle Swarm Optimization		SMPSO
Hybrid Algorithm	Cellular genetic algorithm + differential evolution		CellDE
Stochastic Algorithm	Random Search		RS

KEY RESULTS

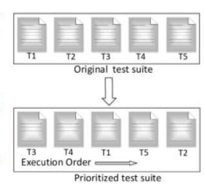
- CellDE, MOCell, NSGA-II and SPEA2 is recommended for the objectives TMP, FPC, FDC and OET when one objective has more importance than the others
- RWGA is recommended when optimizing all the objectives together is sought

Test Prioritization in Product Lines

Content from: Dipesh Pradhan, Shuai Wang, Shaukat Ali, Tao Yue, and Marius Liaaen. "STIPI:Using Search to Prioritize Test Cases Based on Multi-Objectives Derived from Industrial Practice." In *IFIP International Conference on Testing Software and Systems (ICTSS)*, pp. 172-190. Springer, 2016.

In the context of our industrial partner (Cisco), we focused on test case prioritization problem

- Need to execute the important test cases (based on defined criteria) earlier
- Test cases can be prioritized in a large number of ways (n! ways)
- **Problem**: Order the test cases to achieve certain criteria (e.g., fault detection capability) as soon as possible
- Three sets of test cases were used from Cisco: 100, 150, 211



We defined four objectives for our test case prioritization problem in Cisco

- Configuration coverage
  - ✓ Coverage of configuration variables
  - ✓ Coverage of configuration variable values
- Test API coverage
  - ✓ Coverage of test API commands
  - ✓ Coverage of test API parameters
  - ✓ Coverage of test API parameter values
- Status coverage
  - ✓ Coverage of status variables
  - ✓ Coverage of status variable values
- Fault Detection Capability
  - ✓ Rate of failed execution of a test case in a given time period (e.g., a week)
    - ❖ For example, if a test case T_1 failed 4 times after executing 10 times, the fault detection capability of T_1 is 0.4.

We defined two strategies in STIPI for calculating the objectives and integrated with NSGA-II

- Incremental Unique Coverage
  - ✓ Only consider unique elements covered by each subsequent test case
- Position Impact
  - ✓ More importance to test cases scheduled to be executed earlier

$$s_1 = \{T_5, T_1, T_4, T_2, T_3\}$$

$$CVVC_{s_1} = \frac{1\times\frac{5}{5} + 1\times\frac{4}{5} + 1\times\frac{3}{5} + 0\times\frac{2}{5} + 0\times\frac{1}{5}}{3} = 0.8$$

$$s_2 = \{T_1, T_3, T_5, T_2, T_4\}$$

$$CVVC_{s_2} = \frac{1\times\frac{5}{5} + 0\times\frac{4}{5} + 1\times\frac{3}{5} + 0\times\frac{2}{5} + 1\times\frac{1}{5}}{3} = 0.6$$

Test Case	**Configuration**
	protocol
T_1	SIP
T_2	SIP
T_3	SIP
T_4	H323
T_5	H320

Four approaches were selected for empirical evaluation in addition to random search

Approach	Referred As
Greedy	Gr
Giving higher importance to test cases with higher execution position	A1
Combination of modified Average Percentage of Faults Detected (APFD) metric and NSGA-II	A2
Combination of modified APFD with cost metric and NSGA-II	A3

- Arrieta, A., Wang, S., Sagardui, G., Etxeberria, L.: Test Case Prioritization of Configurable Cyber-Physical Systems with Weight-Based Search Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1053-1060. ACM, 2016.
- Rothermel, G., Untch, R.H., Chu, C., Harrold, M.J.: Prioritizing test cases for regression testing. *In IEEE Transactions on Software Engineering (TSE)* vol. 27, no. 10, pp 929-948. IEEE, 2001.
- Elbaum, S., Malishevsky, A., Rothermel, G.: Incorporating varying test costs and fault severities into test case prioritization. In: *Proceedings of International Conference on Software Engineering*, pp. 329–338. IEEE Computer Society, 2001.

Overall STIPI outperformed RS and the selected approaches for three sets of industrial case studies

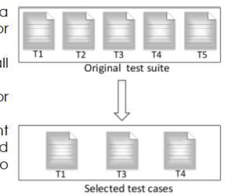
- STIPI outperformed RS for all the evaluation metrics
- STIPI outperformed the four selected approaches for 87.5% of the evaluation metrics

Applications in Maritime Domain

Content from: Dipesh Pradhan, Shuai Wang, Shaukat Ali, and Tao Yue. "Search-Based Cost-Effective Test Case Selection within a Time Budget." In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1085-1092. ACM, 2016.

Based on one of the projects with our industrial partner, we focused on test case selection problem

- Test cases are manual and require a lot of time (e.g., 30 minutes) for execution
- Practically infeasible to execute all the available test cases
- Limited time budget is available for test case execution
- **Problem:** Select the most important test cases (based on defined criteria) with execution time close to the defined time budget



There are four key characteristics for each test case

- Priority
  - ✓ The importance of a test case
  - ✓ Five different values: higher (1.0), high (0.8), medium (0.6), low (0.4), and lower (0.2)
- Probability
  - ✓ The likelihood that a test case may detect a fault
  - ✓ Based on past execution history of the test case
  - ✓ Three different values: high (1.0), medium (0.66), low (0.33)
- Consequence
  - ✓ The impact of a failure of a test case
  - ✓ Five different values: higher (1.0), high (0.8), medium (0.6), low (0.4), and lower (0.2)
- Execution Time

We defined four cost/effectiveness measures

- Mean Priority (MPR)
  - ✓ Average priority of the selected test cases in a solution
- Mean Probability (MPO)
  - ✓ Average probability of the selected test cases in a solution
- Mean Consequence (MC)
  - ✓ Average consequence of the selected test cases in a solution
- Time Difference(TD)
  - ✓ Difference between the time budget and execution cost
  - ✓ Lower value implies better performance
- Preferences should be properly incorporated

An example to show how the different objectives are calculated

Test Case	Priority	Probability	Consequence	Execution Time
T_1	Higher	Medium	High	20 minutes
T_2	High	High	Low	15 minutes
T_3	Lower	Low	Low	20 minutes

- For a solution $\{T_1, T_2\}$ selected within a time budget of 35 minutes
  - ❖ Mean Priority (MPR) = $\frac{1+0.8}{2} = 0.9$
  - ❖ Mean Probability (MPO) = $\frac{0.66+1}{2} = 0.83$
  - ❖ Mean Consequence (MC) = $\frac{0.8+0.4}{2} = 0.6$
  - ❖ Time Difference (TD) = $\mathbf{Nor}\ (35 - 35) = 0$
    - ✓ Note that a lower value for TD implies a better performance

We defined fitness functions for weight-based and Pareto-based search algorithms

- Weight-Based
  - ✓ $FF=1-(MPR \times w_{pr}' + MPO \times w_{po}' + MC \times w_c' + (1-Nor(TD)) \times w_t')$
- Pareto-Based
  - ✓ Five objectives: MPR, MPO, MC, TD, and FF

We evaluated eight search algorithms and two different weight assignment strategies

- Three Weight-Based
  - ✓ Alternating Variable Method (AVM)
  - ✓ Genetic Algorithm (GA)
  - ✓ (1+1) Evolutionary Algorithm (EA)
- Five Pareto-Based
  - ✓ Non-dominated Sorted Genetic Algorithm (NSGA)- II
  - ✓ Strength Pareto Evolutionary Algorithm 2 (SPEA2)
  - ✓ Indicator Based Evolutionary Algorithm (IBEA)
  - ✓ Cellular Genetic Algorithm and Differential Evolution (CellDE)
  - ✓ Multi-objective Cellular (MOCELL) Genetic Algorithm
- Two Weight Assignment Strategies (WASs)
  - ✓ Fixed Weights (FW)
  - ✓ Randomly-Assigned Weights (RAW)

We used two different case studies

- Real World Case Study
  - ✓ 165 test cases
  - ✓ Each test case has four key attributes
    - Priority, probability, consequence, and execution time
- Artificial Problems
  - ✓ Ten test suites with the number of test cases ranging from 100 to 1000 with an increment of 100

Key results for both the real world case study and artificial problems

- All the search algorithms performed significantly better than RS for all the problems
- (1+1) EA with FW performed significantly better than the other combinations of weight-based algorithms and WASs
- SPEA2 with FW significantly outperformed the other Pareto-based search algorithms and WASs
- SPEA2 with FW performed significantly better than (1+1) EA with FW, and therefore for all the selected algorithms and WASs

Test Prioritization in Communication Domain

Content from: S. Wang, S. Ali, T. Tue, Ø. Bakkeli, and M. Liaaen. Enhancing Test Case Prioritization in an Industrial Setting with Resource Awareness and Multi-Objective Search. In The 38th International Conference on Software Engineering (ICSE), Software Engineering in Practice (SEIP) track. pp. 182-191, 2016.

PROBLEM AND CASE STUDIES

- Resource-Aware Multi-Objective Test Case Prioritization Problem
  - ✓ A limited number of available test resources (Hardware)
  - ✓ Several concerns when prioritizing test cases
    - Execution time, fault detection capability
- Case Studies
  - ✓ Industrial: Cisco with one test cycle with 305 test cases and 397 available hardware
  - ✓ 500 artificial problems

OBJECTIVES

- Cost Measure
  - ✓ **TT**: Total time taken by a prioritized set of test cases
    - ✓ Execution time for the test cases
    - ✓ Allocation time for the required test resources
- Three Effectiveness Measures
  - ✓ **PD**: Prioritization density to measure how many test cases are prioritized
  - ✓ **TRU**: Test resource usage to measure to what extent available test resources can be allocated
  - ✓ **FDC**: Fault detection capability
    - ✓ Rate of successful executions in a given time (e.g., a week)

SEARCH ALGORITHMS USED

Algorithm Category			Algorithm
Evolutionary Algorithms (EAs)	GAs	Weight-Based	RWGA
		Sorting-Based	NSGA-II
		Cellular-Based	MOCell
	Strength Pareto EA		SPEA2
	Evolution Strategies		PAES
Swarm Algorithm	Particle Swarm Theory		SMPSO
Hybrid Algorithm	Cellular GA + differential evolution		CellDE

KEY RESULTS

- The search algorithms are cost-effective as compared with Random Ordering in terms of PD, TRU, FDC and TT
- Different algorithms perform the best for different objectives
- RWGA significantly outperforms the others when considering all the objectives together
- The performance of search algorithms is not significantly influenced, when the complexity of the problems increases

Combining Search and Machine Learning

Content from: Dipesh Pradhan, Shuai Wang, Shaukat Ali, Tao Yue, and Marius Liaaen. "REMAP: Using Rule Mining and Multi-Objective Search for Dynamic Test Case Prioritization." Accepted for publication in *IEEE International Conference on Software Testing, Verification and Validation (ICST), 2018.*

There exists internal test case execution relationships

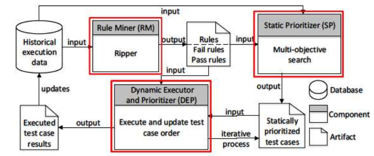
1	2	3	4	5	6	7
T1: Fail	T1: Pass	T1: Pass	T1: Fail	T1: Pass	T1: Fail	T1: Fail
T2: Pass	T2: NE	T2: Pass	T2: Fail	T2: Pass	T2: Fail	T2: Pass
T3: Fail	T3: Pass	T3: Pass	T3: Fail	T3: Pass	T3: NE	T3: Fail
T4: Pass	T4: Fail	T4: Pass	T4: Fail	T4: Pass	T4: Fail	T4: Pass
T5: Pass	T5: Fail	T5: Pass	T5: Pass	T5: Pass	T5: Pass	T5: Pass
T6: Fail	T6: NE	T6: NE	T6: Fail	T6: Fail	T6: Fail	T6: NE

Test Cycle

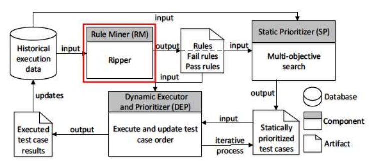
Mine the execution relation among the test cases and classified them into two categories

- Fail rule: An execution relation between two or more test cases if a verdict of a test case(s) is linked to the *fail* verdict of another test case. The goal of this rule is to execute the failing test cases as soon as possible. They are of the form: $(V(T_i)\ AND, \dots, AND\ V(T_k)) \xrightarrow{fail} (V(T_j) = fail) \wedge T_j \notin \{T_i, \dots, T_k\}$
- Pass rule: An execution relation between two or more test cases if a verdict of a test case(s) is linked to the *pass* verdict of another test case. The goal of this rule is to execute the passing test cases later. They have the following form: $(V(T_i)\ AND, \dots, AND\ V(T_k)) \xrightarrow{pass} (V(T_j) = pass) \wedge T_j \notin \{T_i, \dots, T_k\}$

THE PROPOSED APPROACH (REMAP) CONSISTS OF THREE KEY COMPONENTS FOR DYNAMIC TEST CASE PRIORITIZATION



RULE MINER (RM)



THE RULE MINER IS USED TO INFER **FAIL** AND **PASS** RULES FROM THE HISTORICAL EXECUTION RESULTS

- Fail Rule
  - If a verdict of a test case(s) in the **rule antecedent** is linked to the *fail* verdict of another test case in the **rule consequent**
  - $(V(T_i)\ AND, \dots, AND\ V(T_k))$ —fail—> $(V(T_j) = fail) \wedge T_j \notin \{T_i, \dots, T_k\}$

1	2	3	4	5	6	7
T1: Fail	T1: Pass	T1: Pass	T1: Fail	T1: Pass	T1: Fail	T1: Fail
T2: Pass	T2: NE	T2: Pass	T2: Fail	T2: Pass	T2: Fail	T2: Pass
T3: Fail	T3: Pass	T3: Pass	T3: Fail	T3: Pass	T3: NE	T3: Fail
T4: Pass	T4: Fail	T4: Pass	T4: Fail	T4: Pass	T4: Fail	T4: Pass
T5: Pass	T5: Fail	T5: Pass	T5: Pass	T5: Pass	T5: Pass	T5: Pass
T6: Fail	T6: NE	T6: NE	T6: Fail	T6: Fail	T6: Fail	T6: NE

Test Cycle

THE RULE MINER IS USED TO INFER **FAIL** AND **PASS** RULES FROM THE HISTORICAL EXECUTION RESULTS

- Pass Rule
  - If a verdict of a test case(s) in the **rule antecedent** is linked to the *pass* verdict of another test case in the **rule consequent**
  - $(V(T_i)\ AND, \ldots, AND\ V(T_k))$ —pass—> $(V(T_j)$ = pass$) \wedge T_j \notin \{T_i, \ldots, T_k\}$

1	2	3	4	5	6	7
T1: Fail	T1: Pass	T1: Pass	T1: Fail	T1: Pass	T1: Fail	T1: Fail
T2: Pass	T2: NE	T2: Pass	T2: Fail	T2: Pass	T2: Fail	T2: Pass
T3: Fail	T3: Pass	T3: Pass	T3: Fail	T3: Pass	T3: NE	T3: Fail
T4: Pass	T4: Fail	T4: Pass	T4: Fail	T4: Pass	T4: Fail	T4: Pass
T5: Pass	T5: Fail	T5: Pass	T5: Pass	T5: Pass	T5: Pass	T5: Pass
T6: Fail	T6: NE	T6: NE	T6: Fail	T6: Fail	T6: Fail	T6: NE

Test Cycle

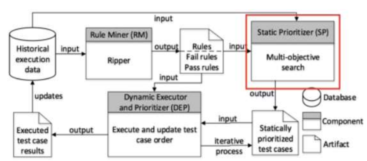
THE RULE MINER LEVERAGES THE RIPPER ALGORITHM TO MINE THE **FAIL** AND **PASS** RULES

- To mine *fail* rules and *pass* rules
  - Repeated Incremental Pruning to Produce Error Reduction (RIPPER)
  - Input
    - ✓ A test suite with *n* test cases
    - ✓ Historical test execution data for the *n* test cases

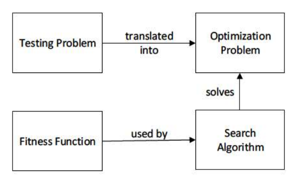
1	2	3	4	5	6	7
T1: Fail	T1: Pass	T1: Pass	T1: Fail	T1: Pass	T1: Fail	T1: Fail
T2: Pass	T2: NE	T2: Pass	T2: Fail	T2: Pass	T2: Fail	T2: Pass
T3: Fail	T3: Pass	T3: Pass	T3: Fail	T3: Pass	T3: NE	T3: Fail
T4: Pass	T4: Fail	T4: Pass	T4: Fail	T4: Pass	T4: Fail	T4: Pass
T5: Pass	T5: Fail	T5: Pass	T5: Pass	T5: Pass	T5: Pass	T5: Pass
T6: Fail	T6: NE	T6: NE	T6: Fail	T6: Fail	T6: Fail	T6: NE

Test Cycle

STATIC PRIORITIZER (SP)



THE STATIC PRIORITIZER SOLVES THE MULTI-OBJECTIVE TEST PRIORITIZATION PROBLEM USING A SEARCH ALGORITHM



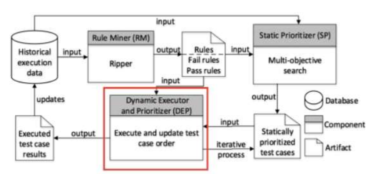
THE STATIC PRIORITIZER MAXIMIZES TWO OBJECTIVES FOR BLACK-BOX TEST CASE PRIORITIZATION

- Objectives
  - Fault Detection Capability (*FDC*)
    - Rate of failed execution of a test case in a given time period
  - Test Case Reliance Score (*TRS*)
    - *TRS* for a test case T_i is defined as the unique number of test cases whose results can be predicted by T_i using the defined *fail* rules and *pass* rules

THE STATIC PRIORITIZER INTEGRATES THE OBJECTIVES INTO NSGA-II AND PRODUCES A SET OF NON-DOMINATED SOLUTIONS

- Static prioritizer uses Non-dominated Sorting Genetic Algorithm (NSGA)-II
- Static prioritizer produces a set of non-dominated solutions based on Pareto-optimality theory
  - Solution s_1 dominates solution s_2 iff:
    - s_1 is better than s_2 in at least one objective
    - For all the other objective(s), s_1 is not worse than s_2

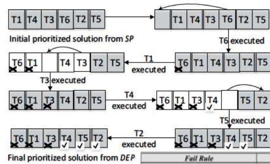
DYNAMIC EXECUTOR AND PRIORITIZER



DYNAMIC EXECUTOR AND PRIORITIZER EXECUTES AND DYNAMICALLY PRIORITIZES THE TEST CASES

- The objective of DEP is to:
  - Execute the statically prioritized test cases from static prioritizer
  - Dynamically update the test cases execution order based on:
    - Runtime execution results
    - Mined *fail* rules and *pass* rules from the rule miner
- DEP moves the test cases that failed 100% of the time in the historical execution data to the first position(s)

TEST CASES ARE PRIORITIZED DYNAMICALLY BASED ON THE RUNTIME EXECUTION RESULTS OF THE RELATED TEST CASES



Fail Rule	Pass Rule
1. $(V(T_1) = fail) \xrightarrow{fail} (V(T_3) = fail)$	5. $(V(T_2) = pass) \xrightarrow{pass} (V(T_5) = pass)$
2. $(V(T_2) = fail) \xrightarrow{fail} (V(T_4) = fail)$	6. $(V(T_5) = pass) \xrightarrow{pass} (V(T_1) = pass)$
3. $(V(T_3) = fail) \xrightarrow{fail} (V(T_1) = fail)$	7. $(V(T_4) = pass) \xrightarrow{pass} (V(T_2) = pass)$
4. $(V(T_4) = fail) \xrightarrow{fail} (V(T_2) = fail)$	

[simula . research laboratory] 81

We used five different case studies

CS	Data Set	#Test Cases	#Test Cycles	#Verdicts
CS_1	Industrial Data Set1	60	8,322	296,042
CS_2	Industrial Data Set2	624	6,302	149,039
CS_3	ABB Paint Control	89	351	25,497
CS_4	ABB IOF/ROL	1,941	315	32,162
CS_5	Google GSDTSR	5,555	335	1,253,464

[simula . research laboratory] 82

Key results for the five case studies

- REMAP significantly outperformed random ordering, greedy, static, and rule-based approaches
- REMAP managed to achieve on average 18% higher Average Percentage of Fault Detected(*APFD*)

[simula . research laboratory] 83

TOOLS/SOURCE CODE

- Available at
  - ✓ http://zen-tools.com/
- AVMf Framework
  - ✓ http://avmframework.org/

[simula . research laboratory] 84