The Impact of Source Test Case Selection on the Effectiveness of Metamorphic Testing

A.C. Barus, T.Y. Chen, F.-C. Kuo, H. Liu, H.W. Schmidt

huai.liu@rmit.edu.au

Outline

- Motivation
- Background
- Experiment
- Result
- Conclusion

Motivation

- Implementation of metamorphic testing (MT)
  - Metamorphic relations (MRs)
  - Source test cases
  - Follow-up test cases
- Effectiveness of metamorphic testing (MT)
  - Good MRs
  - Good test cases

Motivation

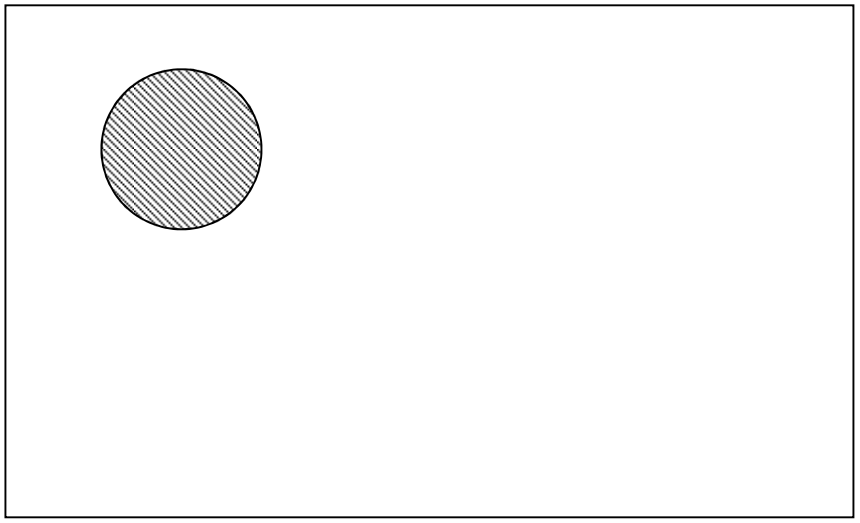
- Generation of source test cases
  - Pure random
  - Special values

Motivation

- Enhancement of random testing
  - Adaptive random testing (ART)
  - Test case diversity
- Source test case generation using ART

Background

- ART
  - Contiguous failure regions → contiguous non-failure regions
  - Given a test case t does not reveal failure, the input adjacent (similar) to t is very likely to be non-failure-causing
  - An even spread of random test cases will enhance the effectiveness



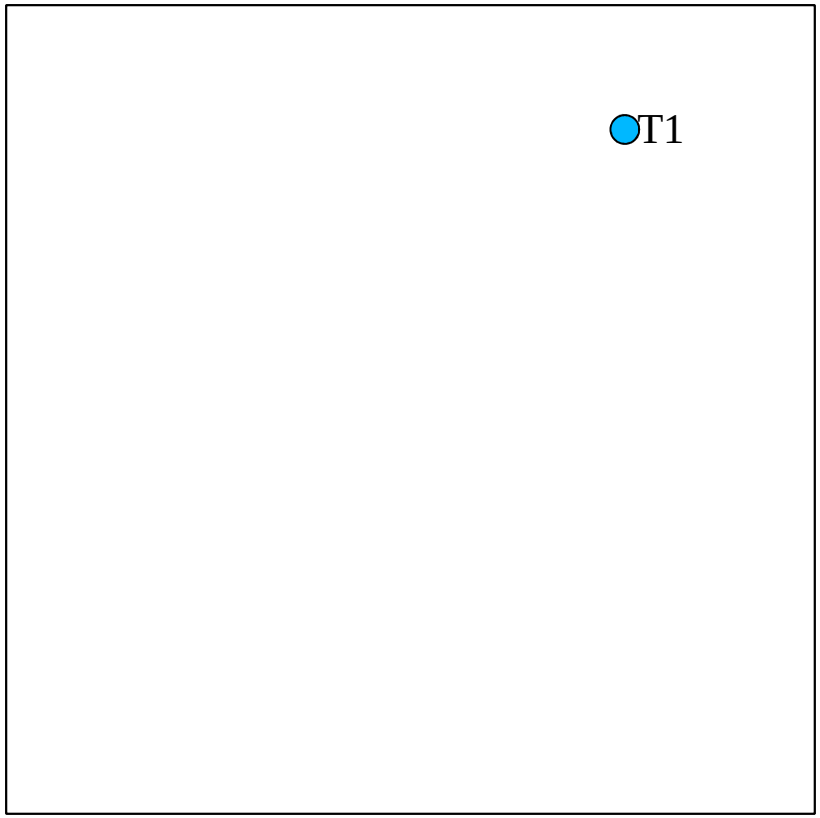
Background

- Even spread, even distribution, diversity, variant
  - Measurement of difference/similarity/distance
  - Algorithms for even spreading

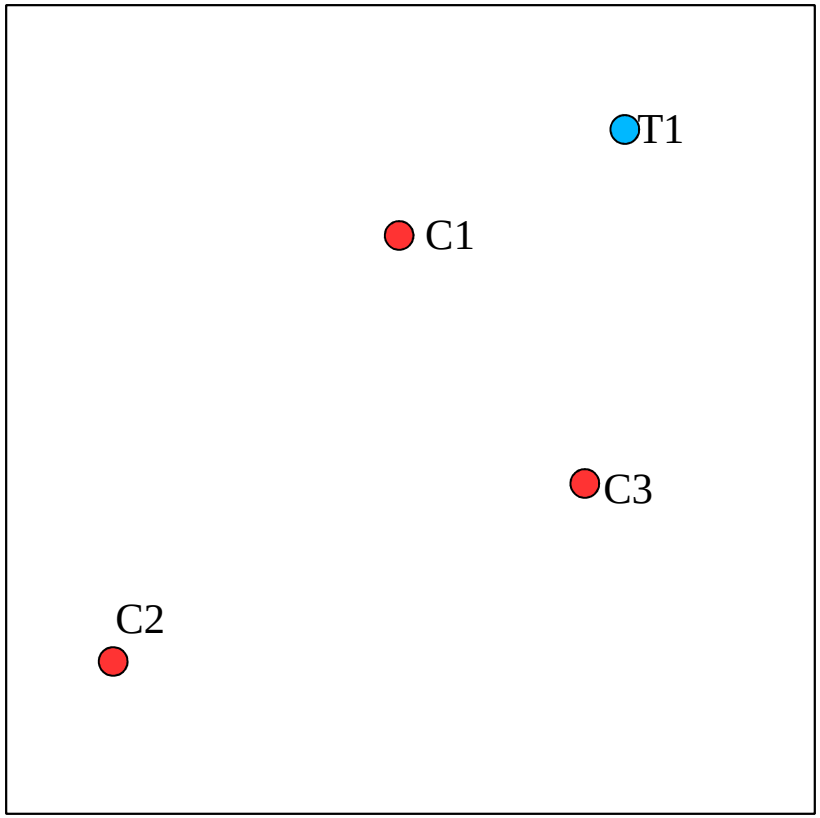
Background

- FSCS-ART: One algorithm for even spreading
  - A set of executed test cases
  - A set of candidates
  - The best candidate → next test case

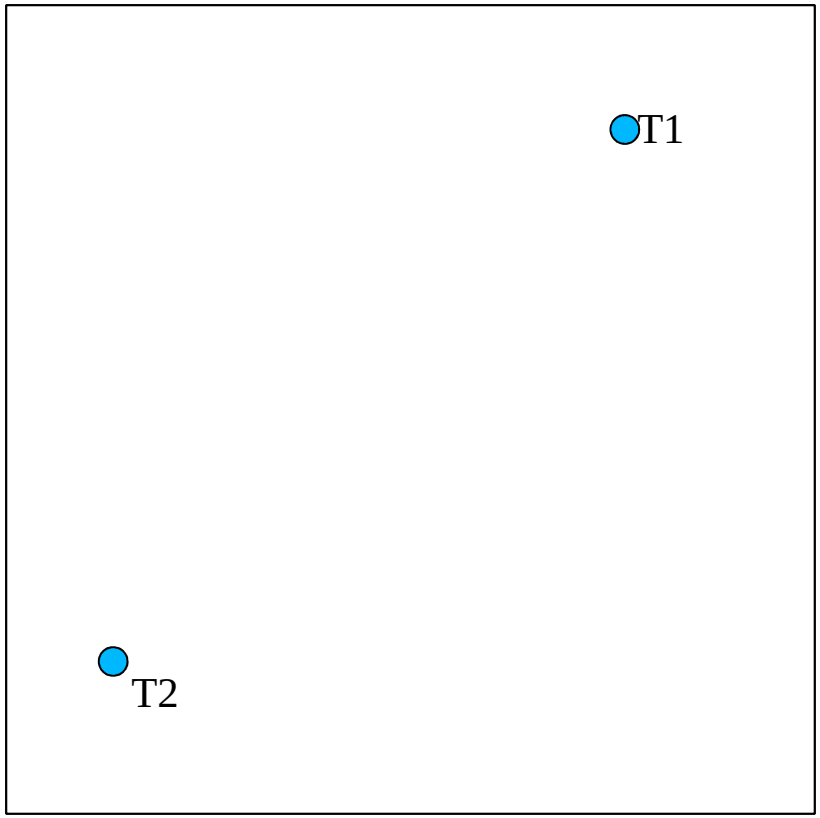
Background



Background



Background



Background

- How to select the best candidate
  - max-min:
    - d_{nn}: distance to nearest neighbor
    - The largest d_{nn}
  - max-sum
    - d_{sum}: sum of distances to all executed test cases
    - The largest d_{sum}

Background

- Distance measurement
  - Numeric: Euclidean distance, discrepancy, dispersion.
  - Non-numeric: OO distance, model-based
  - More general: category partition based

Background

- Category partition method
  - Identify input parameters and environment variables that affect the execution behaviour
  - Find categories of information that characterize each parameter and variable
  - Partition each category into choices
  - Determine the constraints among choices
  - Construct test frames
  - Generate test case from test frame

Background

- For a given concrete input, we identify its relevant test frame.
- We take two program inputs, determine their categories and choices, and use this information to calculate the distance between them, with a greater distance representing the situation in which the two inputs are more dissimilar.
- Given two program inputs *x* and *y*, our distance measure is a count of the number of categories in which *x* and *y* have different choices.

Background

- A transaction processing system handles a large range of monetary quantities (for instance, it may deal with cash and credit transactions, and the transfer of items from a stock inventory).

Category	Choice
Unit type	Cheque
	Credit
	Inventory item
Customer type	Business
	Personal
	Government
	Other
Status	Accepted
	Rejected

Background

- Three example inputs

Input	Processed Transaction	Category and Choice
x	A cleared cheque payment of \$123.45 from Anycorp, a business customer.	Unit type:Cheque
		Customer type:Business
		Status:Accepted
y	A credit card payment of \$543.21 from Mr. Fred Phisher, a personal customer whose dubious identity leads to the payment being rejected.	Unit type:Credit
		Customer type:Personal
		Status:Rejected
z	The dispatch of 12 widgets from stock to Othercorp, a business customer. The order is accepted.	Unit type:Inventory item
		Customer type:Business
		Status:Accepted

Background

- Distance calculation

Between pair of	DP	DA	|DA|
(x, y)	Unit type:Cheque	Unit type	3
	Unit type:Credit		
	Customer type:Business	Customer type	
	Customer type:Personal		
	Status:Accepted	Status	
	Status:Rejected		
(x, z)	Unit type:Cheque	Unit type	1
	Unit type:Inventory Item		
(y, z)	Unit type:Credit	Unit type	3
	Unit type:Inventory Item		
	Customer type:Personal	Customer type	
	Customer type:Business		
	Status:Rejected	Status	
	Status:Accepted		

Experiment

- Research question
  - Can the use of ART in the source test case selection improve the effectiveness of MT?
- Object programs
  - grep: regular expression, 6 MRs, 14 categories
  - printtokens/printtokens2: 3 MRs, 28 categories
  - schedule/schedule2: 3 MRs, 34 categories
  - replace: 3 MRs,24 categories

Experiment

- Techniques under study
  - RT
  - ART with max-min
  - ART with max-sum
  - Aging: a technique for reducing overhead of ART
- Effectiveness metric
  - F-measure
  - Expected number of test cases to detect the first failure

Result

Program	Technique	RT	ART *max-min with aging*	ART *max-sum with aging*
grep	RT	N/A	**57**	26
	ART *max-min with aging*	**12**	N/A	11
	ART *max-sum with aging*	43	58	N/A
printtokens	RT	N/A	9	9
	ART *max-min with aging*	0	N/A	**9**
	ART *max-sum with aging*	0	**0**	N/A
printtokens-2	RT	N/A	20	**20**
	ART *max-min with aging*	8	N/A	17
	ART *max-sum with aging*	**8**	11	N/A
schedule	RT	N/A	3	3
	ART *max-min with aging*	7	N/A	2
	ART *max-sum with aging*	7	8	N/A
schedule-2	RT	N/A	**19**	**20**
	ART *max-min with aging*	**1**	N/A	**18**
	ART *max-sum with aging*	**0**	**2**	N/A
replace	RT	N/A	**16**	17
	ART *max-min with aging*	**7**	N/A	10
	ART *max-sum with aging*	6	13	N/A

Result

- ART max-sum significantly outperformed RT
  - printtokens, printtokens2, schedule2
- ART max-min significantly outperformed RT
  - grep, schedule2, replace
- ART max-sum significantly outperformed ART max-min
  - schedule2
- RT could not significantly outperform ART

Conclusion

- Improvement of MT effectiveness
- ART in source test case selection
- Fewer test cases required to detect the first failure

Thank you