Diversity in Test Suite Optimization

Annibale Panichella – PhD Candidate

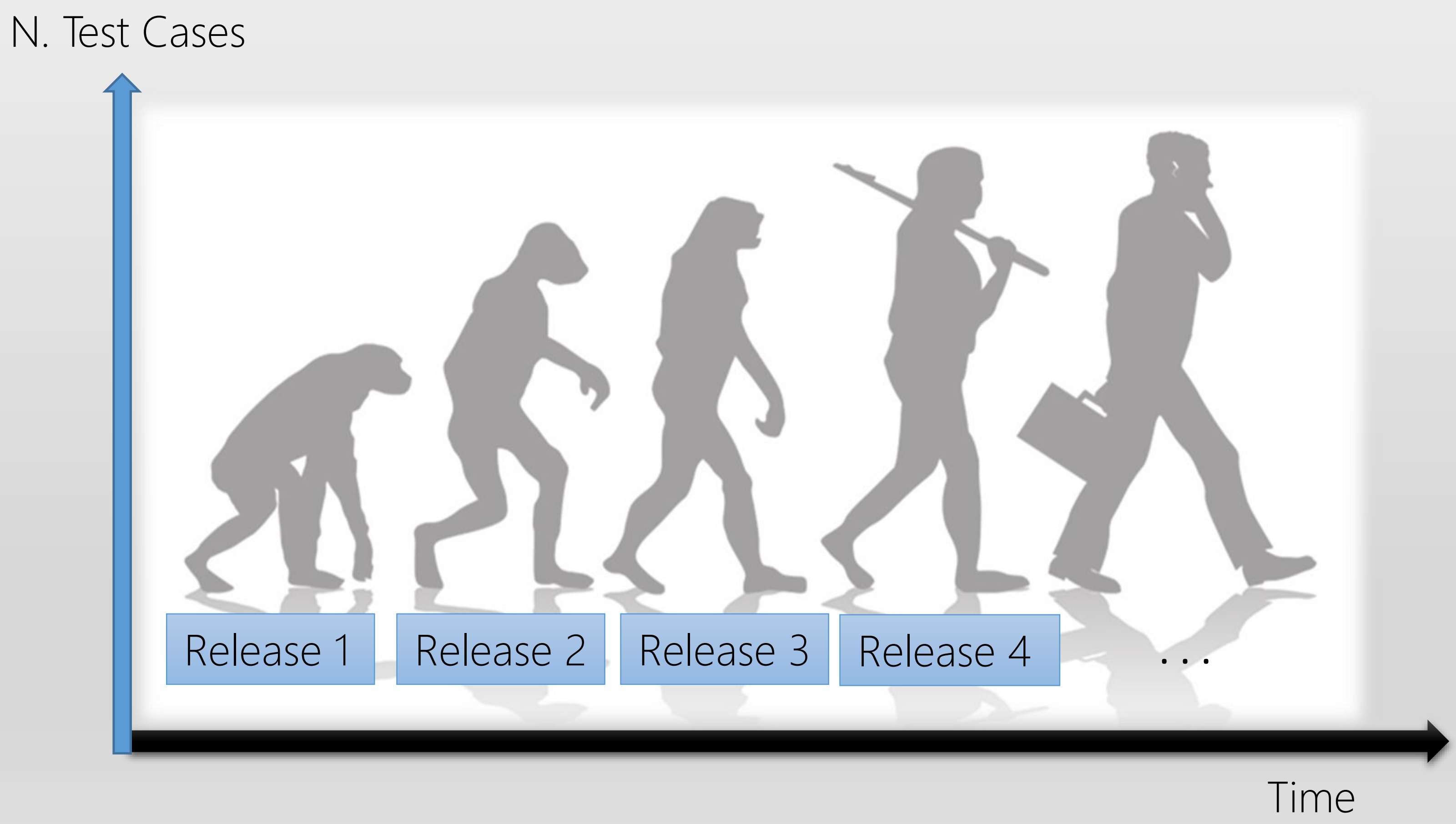
apanichella@unisa.it

Software Engineering Lab , University of Salerno, Italy

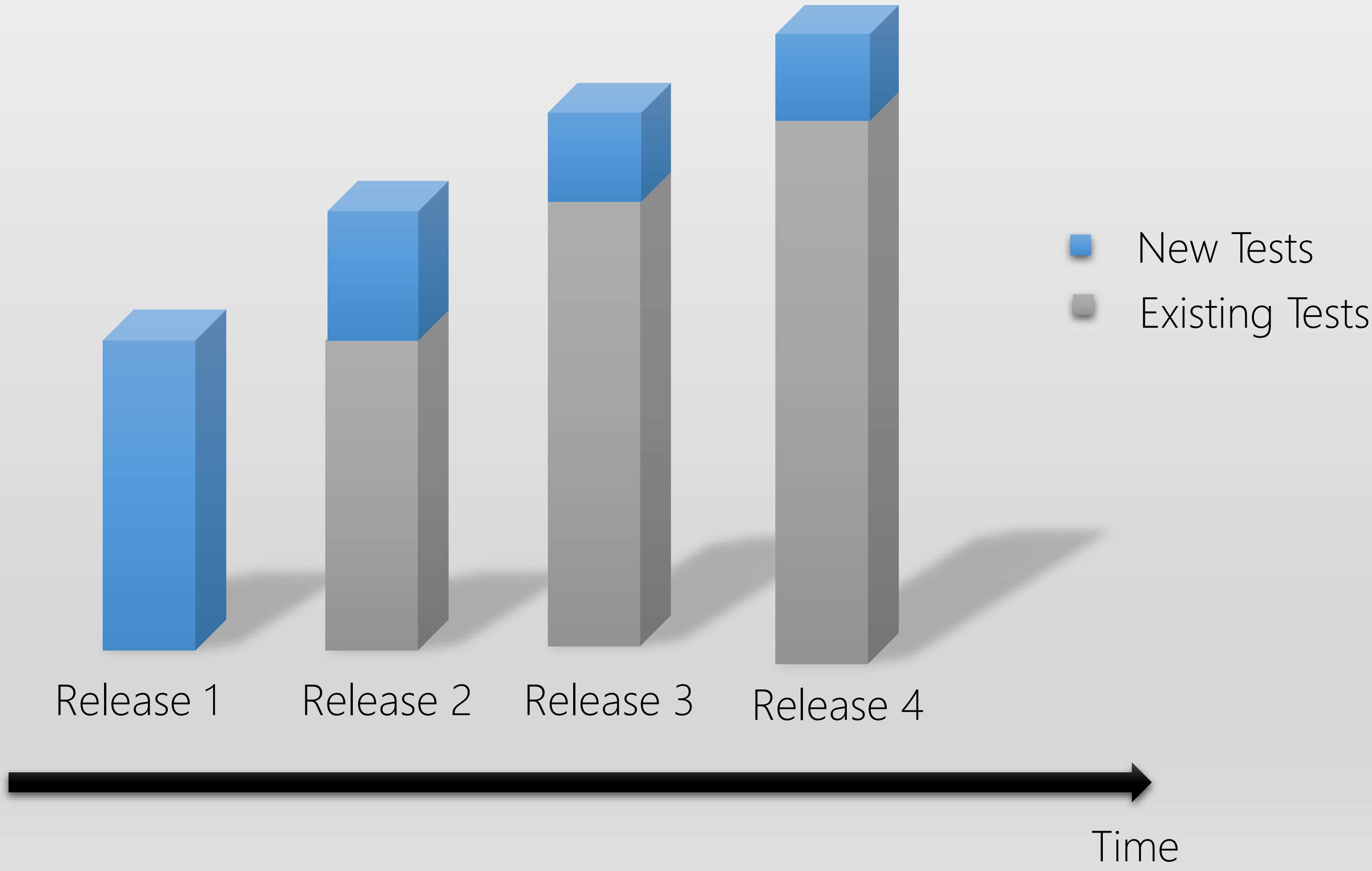
Andrea De Lucia
University of Salerno

Rocco Oliveto
University of Molise

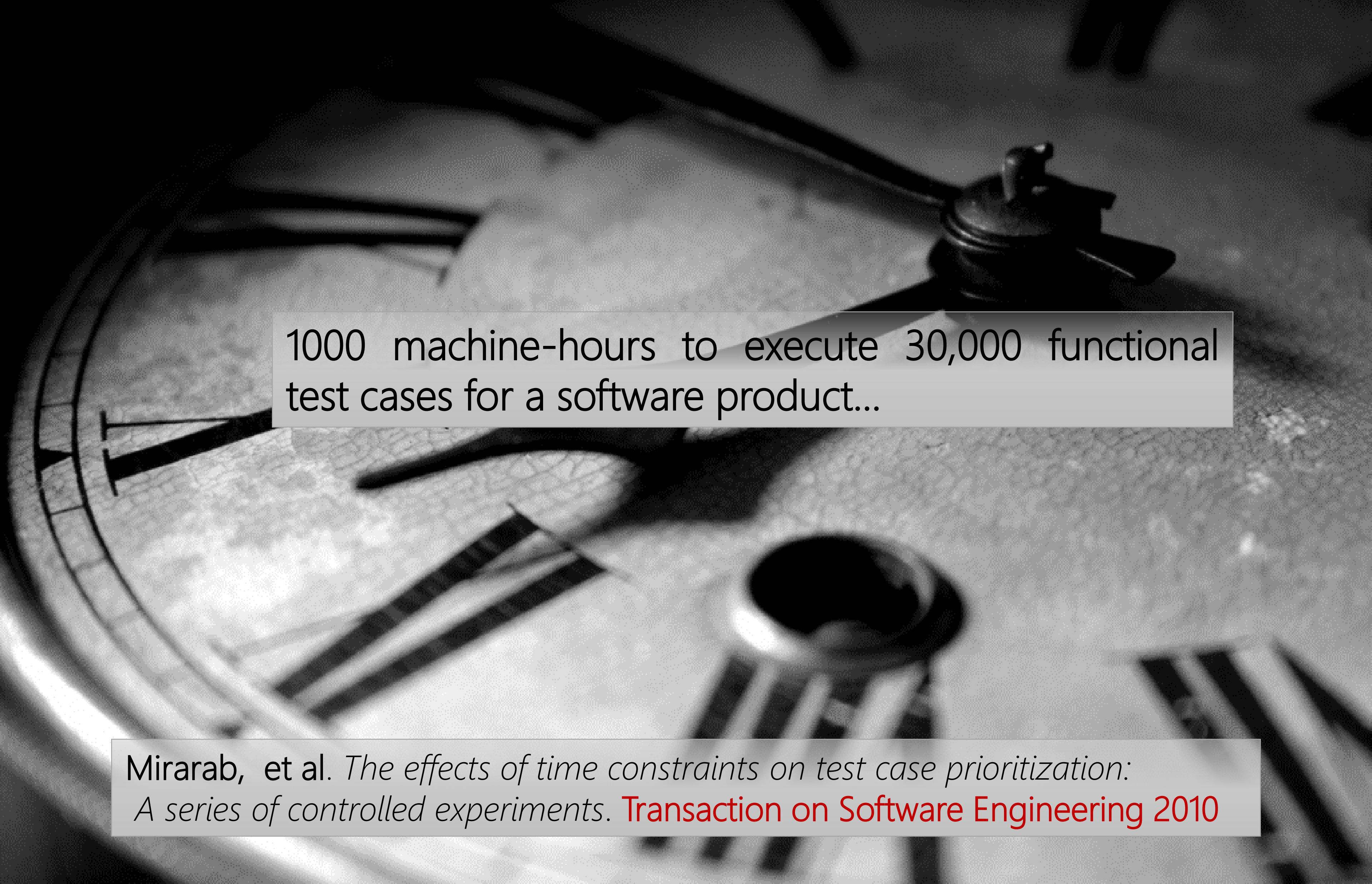
Software Evolution



Amount of regression testing



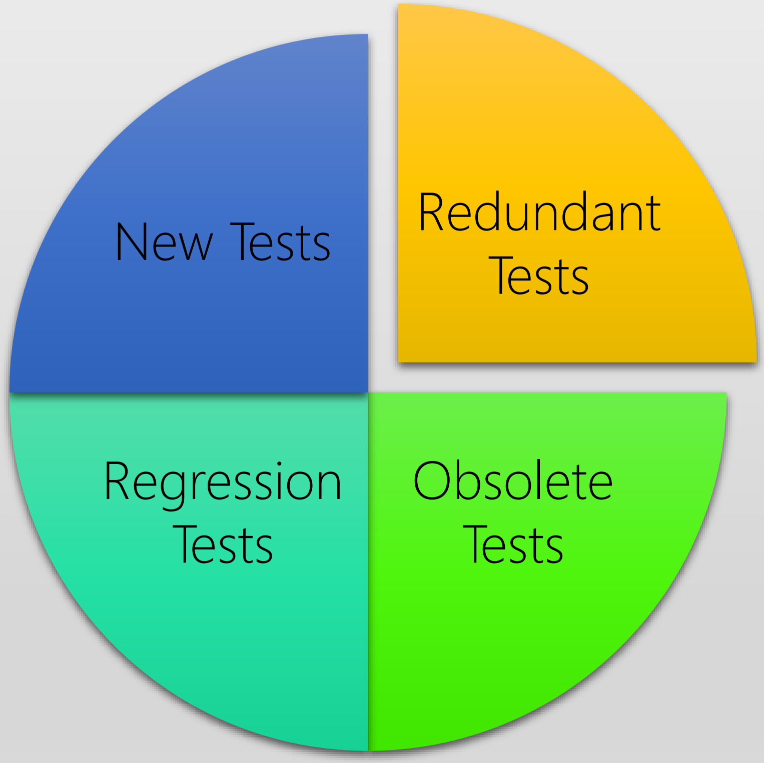
Regression Testing is time consuming



1000 machine-hours to execute 30,000 functional test cases for a software product…

Mirarab, et al. *The effects of time constraints on test case prioritization: A series of controlled experiments*. Transaction on Software Engineering 2010

Test Cases Redundancy



Reduce the Test Suite Size
Reduce the Test Suite Cost
Preserve the effectiveness

Coverage Criteria


A Methodology for Controlling the Size of a Test Suite

MARY JEAN HARROLD
Clemson University

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 22, NO. 8, AUGUST 1996

Analyzing Regression Test Selection Techniques

Gregg Rothermel, *Member, IEEE* and Mary Jean Harrold, *Member, IEEE*

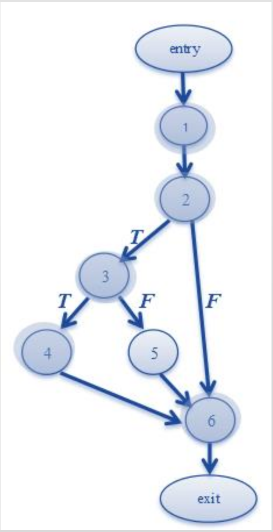
Abstract—Regression testing is a necessary but expensive maintenance activity aimed at showing that code has not been adversely affected by changes.

1 INTRODUCTION

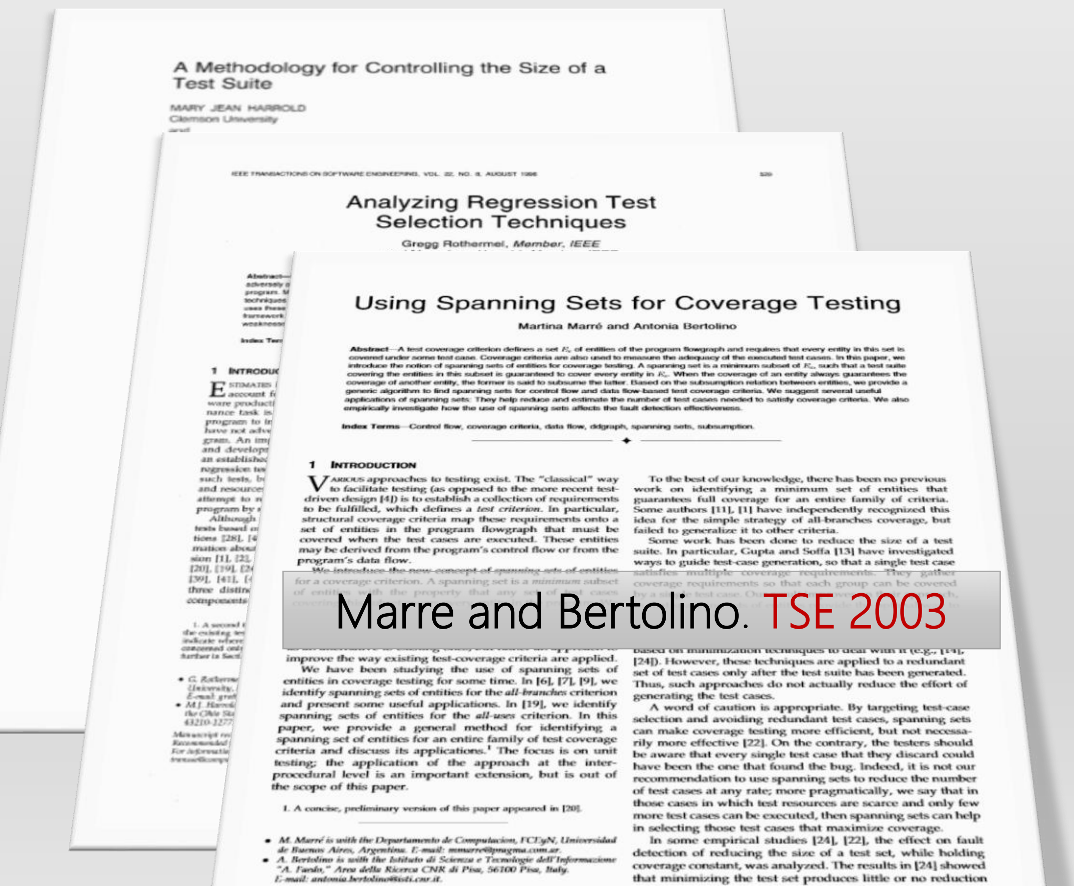
Code coverage

- Statement coverage
- Block coverage

	Program
	integer i, j, k
1.	read i, j, k
2.	if(i < j)
3.	if(j < k)
4.	i = k;
	else
5.	k = i;
	endif
	endif
6.	print i, j, k
	end



Coverage Criteria



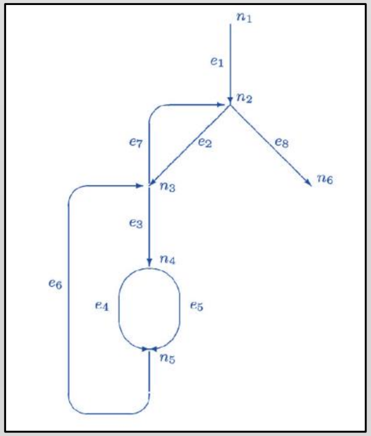
Marre and Bertolino. TSE 2003

Decision-To-Decision Graph

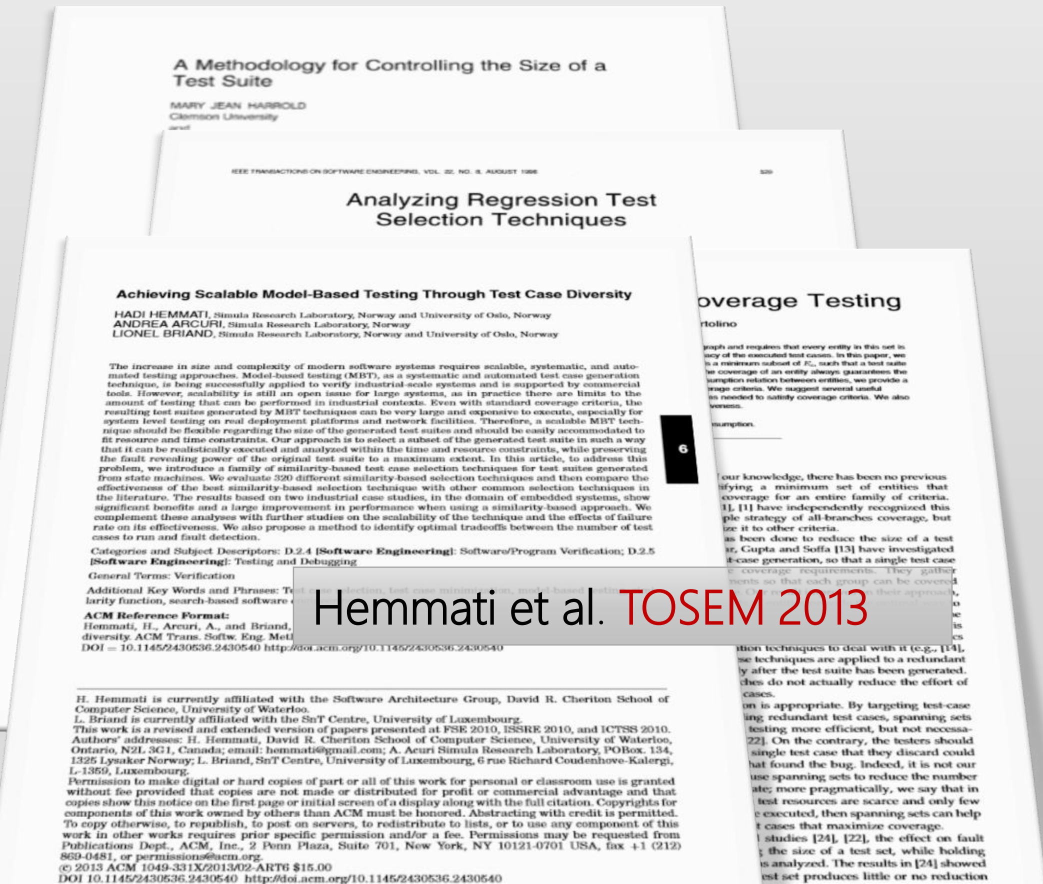
- All paths coverage
- All branches coverage
- All uses
- …

```
void sort(a, n) /* SORT Program */
  int a[];
  int n;
{
   int sortupto;
   int maxpos;
   int mymax;
   int index;

   sortupto = 1;
   maxpos = 1;
   while (sortupto < n) {
      mymax = a[sortupto];
      index = sortupto + 1;
      while (index <= n) {
         if (a[index] > mymax) {
            mymax = a[index];
            maxpos = index;
         }
         index = index + 1;
      }
      index = a[sortupto];
      a[sortupto] = mymax;
      a[maxpos] = index;
      sortupto = sortupto + 1;
   }
}
```

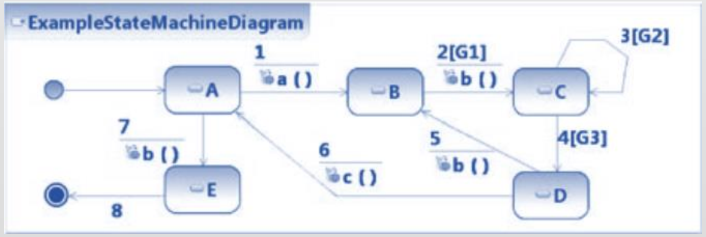


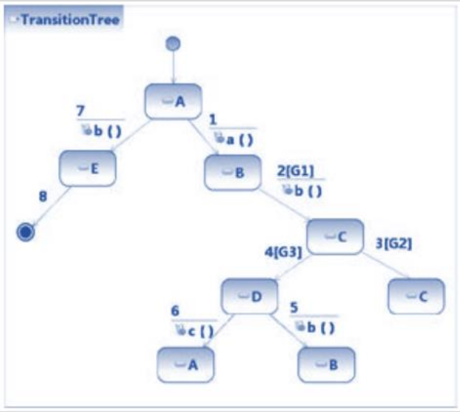
Coverage Criteria



Model-Based coverage

- State Machine transitions
- Transition Tree





Test Suite Optimization

Redundant Test Cases

Input:

- Program $P = \{e_1, e_2, ..., e_n\}$
- Test Cases covering parts of P

$$T = \{t_1, t_2, ..., t_n\}$$

Problem:

Finding the minimal sub-set

$T^* \subseteq T$ such that $\bigcup_{t \in T^*} t(P) = P$

Test Suite Optimization

Redundant Test Cases

Input:
- Program $P=\{e_1,e_2,\ldots,e_n\}$
- Test Cases covering parts of P

$$T=\{t_1,t_2,\ldots,t_n\}$$

Problem:
Finding the minimal sub-set $T^* \subseteq T$ such that $\bigcup_{t \in T^*} t(P) = P$

Hitting set

Input:
- Set $P=\{e_1,e_2,\ldots,e_n\}$
- A collection of sub-sets of P

$$T=\{t_1,t_2,\ldots,t_n\}$$

Problem:
Finding the minimal sub-set $T^* \subseteq T$ such that $\bigcup_{t \in T^*} t_i = P$

Test Suite Optimization

The problem is NP-hard

Redunda… Test Cases

Input:
- Program
- Test Case…

T…

Problem:
Finding the minimal sub-set
$T^* \subseteq T$ such that $\bigcup_{t \in T^*} t(P) = P$

Hitting set

Input:
…$\{e_1, e_2, \ldots, e_n\}$
…s of P

Finding t… …b-set
$T^* \subseteq T$ such that $\bigcup_{t \in T^*} t_i = P$

Search

100 Test Cases

N. of possible subsets $2^{100} \approx 1{,}26 \cdot 10^{30} >$

$4.354 \pm 0.012 \cdot 10^{17}$ seconds Age of the Universe

Additional Greedy Algorithm

	Greedy Algorithm (P, T)	
(1)	C ←Ø	covered statements of P
(2)	S ←Ø	set of selected test cases
(2)	**repeat**	
(3)	j← max {Tj – (C ∩ Tj)}	
(4)	C = C U Tj	
(5)	Remove Tj from T	
	S = S U {Tj}	
(6)	**until** C=P	

1) Start with the test case having the highest coverage
2) Iteratively add the most distant test case
3) Perform step 1-2 until max coverage is reached

Additional Greedy Algorithm

	Greedy Algorithm (P, T)	
(1)	C ←∅	covered statements of P
(2)	S ←∅	set of selected test cases
(2)	**repeat**	
(3)	j← max {Tj – (C ∩ Tj)}	
(4)	C = C U Tj	
(5)	Remove Tj from T	
	S = S U {Tj}	
(6)	**until** C=P	

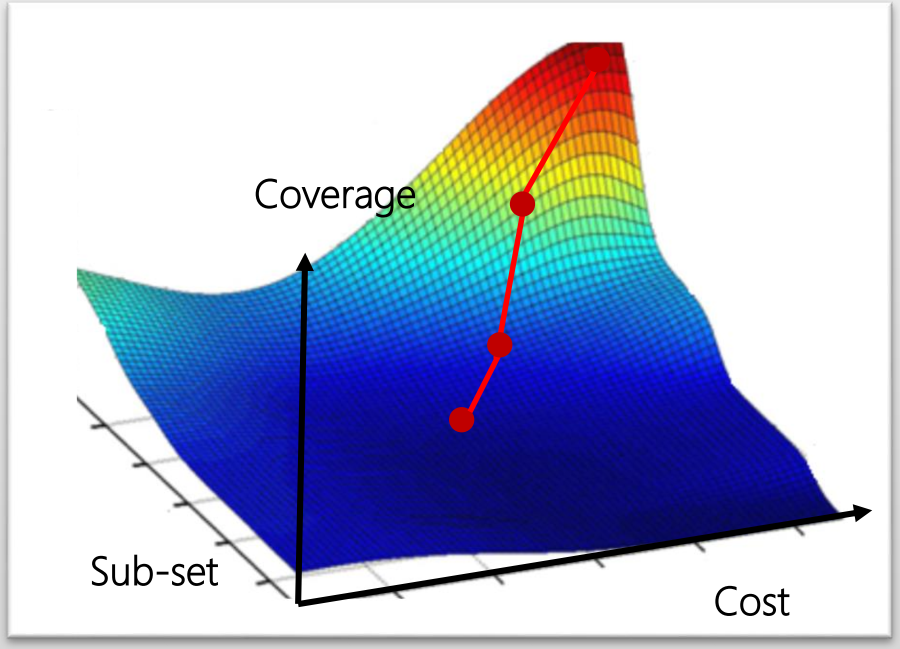
1) Start with the test case having the highest coverage
2) Iteratively add the most distant test case
3) Perform step 1-2 until max coverage is reached

Greedy step = $T_j - (C \cap T_j) \propto (C \cup T_j) - (C \cap T_j) \propto$ Jaccard Distance

Greedy step is proportional to the Jaccard Distance

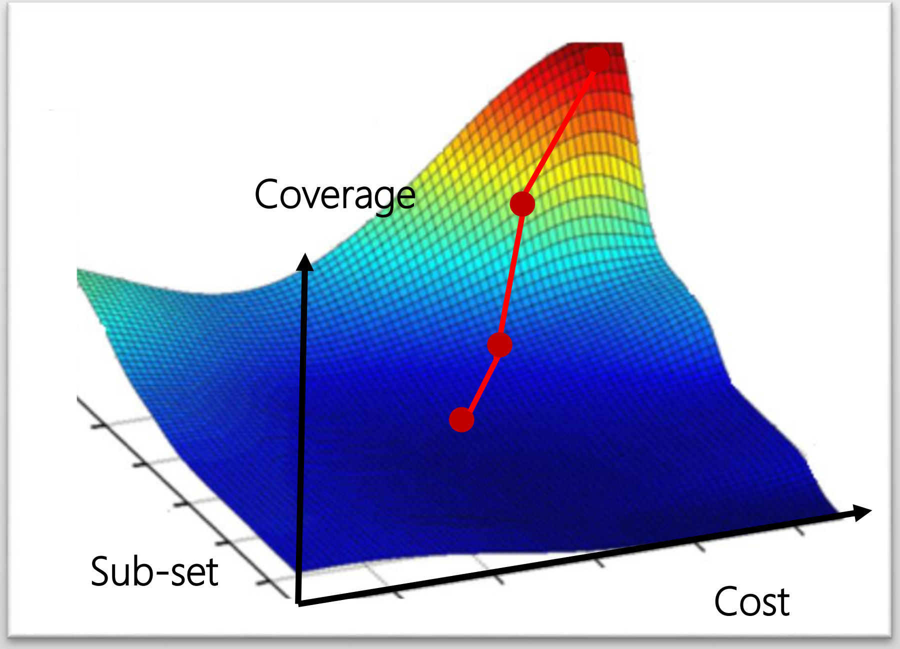
Additional Greedy Algorithm

Ideal Problem

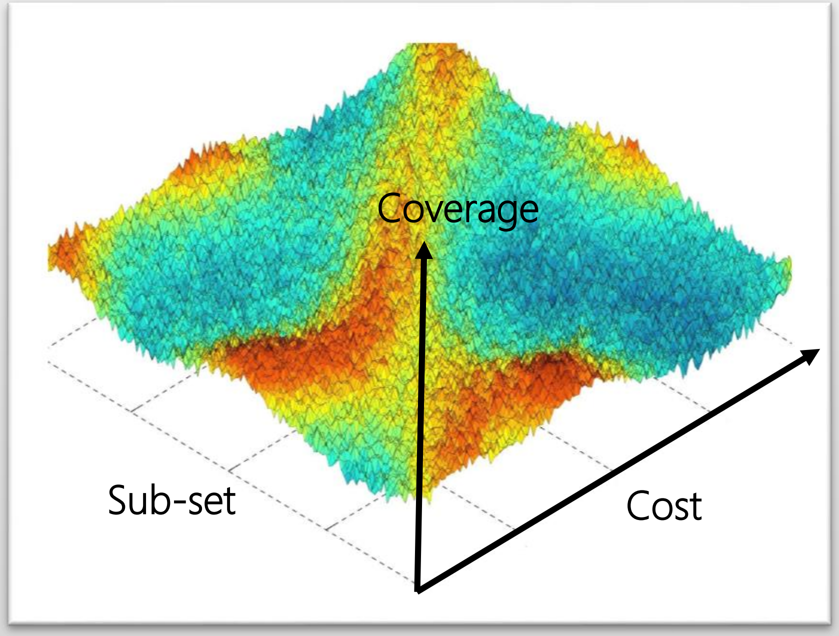


Additional Greedy Algorithm

Ideal Problem



Real Problem



What is the best search algorithm?

Achieving Scalable Model-Based Testing Through Test Case Diversity

HADI HEMMATI, Simula Research Laboratory, Norway and University of Oslo, Norway
ANDREA ARCURI, Simula Research Laboratory, Norway
LIONEL BRIAND, Simula Research Laboratory, Norway and University of Oslo, Norway

The increase in size and complexity of modern software systems requires scalable, systematic, and automated testing approaches. Model-based testing (MBT), as a systematic and automated test case generation technique, is being successfully applied to verify industrial-scale systems and is supported by commercial tools. However, scalability is still an open issue for large systems, as in practice there are limits to the amount of testing that can be performed in industrial contexts. Even with standard coverage criteria, the resulting test suites generated by MBT techniques can be very large and expensive to execute, especially for system level testing on real deployment platforms and network facilities. Therefore, a scalable MBT technique should be flexible regarding the size of the generated test suites and should be easily accommodated to fit resource and time constraints. Our approach is to select a subset of the generated test suite in such a way that it can be realistically executed and analyzed within the time and resource constraints, while preserving the fault revealing power of the original test suite to a maximum extent. In this article, to address this problem, we introduce a family of similarity-based test case selection techniques for test suites generated from state machines. We evaluate 320 different similarity-based selection techniques and then compare the effectiveness of the best similarity-based selection technique with other common selection techniques in the literature. The results based on two industrial case studies, in the domain of embedded systems, show significant benefits and a large improvement in performance when using a similarity-based approach. We complement these analyses with further studies on the scalability of the technique and the effects of failure rate on its effectiveness. We also propose a method to identify optimal tradeoffs between the number of test cases to run and fault detection.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Software/Program Verification; D.2.5 [**Software Engineering**]: Testing and Debugging

General Terms: Verification

Additional Key Words and Phrases: Test case selection, test case minimization, model-based testing, similarity function, search-based software engineering

ACM Reference Format:
Hemmati, H., Arcuri, A., and Briand, L. 2013. Achieving scalable model-based testing through test case diversity. ACM Trans. Softw. Eng. Methodol. 22, 1, Article 6 (February 2013), 42 pages.
DOI = 10.1145/2430536.2430540 http://doi.acm.org/10.1145/2430536.2430540

H. Hemmati is currently affiliated with the Software Architecture Group, David R. Cheriton School of Computer Science, University of Waterloo.

L-1359, Luxembourg.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2013 ACM 1049-331X/2013/02-ART6 $15.00
DOI 10.1145/2430536.2430540 http://doi.acm.org/10.1145/2430536.2430540

ACM Transactions on Software Engineering and Methodology, Vol. 22, No. 1, Article 6, Pub. date: February 2013.

Hemmati et al. TOSEM 2013

1) Model Driven Testing
   - Model-based coverage
   - Abstract Test Cases

2) Search Algorithms
   - Greedy Algorithm
   - Random Search Algorithm
   - Hill climbing
   - Evolutionary Algorithms

3) Similarity functions
   - Set distance
   - Hamming distance
   - Jaccard distance
   - Clustering distance

What is the best search algorithm?

Achieving Scalable Model-Based Testing Through Test Case Diversity

HADI HEMMATI, Simula Research Laboratory, Norway and University of Oslo, Norway
ANDREA ARCURI, Simula Research Laboratory, Norway
LIONEL BRIAND, Simula Research Laboratory, Norway and University of Oslo, Norway

The increase in size and complexity of modern software systems requires scalable, systematic, and automated testing approaches. Model-based testing (MBT), as a systematic and automated test case generation technique, is being successfully applied to verify industrial-scale systems and is supported by commercial tools. However, scalability is still an open issue for large systems, as
amount of testing that can be performed in industrial contexts. Even
resulting test suites generated by MBT techniques can be very large an
system level testing on real deployment platforms and network facili
nique should be flexible regarding the size of the generated test suites
fit resource and time constraints. Our approach is to select a subset of
that it can be realistically executed and analyzed within the time and
the fault revealing power of the original test suite to a maximum
problem, we introduce a family of similarity-based test case selectio
from state machines. We evaluate 320 different similarity-based sele
effectiveness of the best similarity-based selection technique with
the literature. The results based on two industrial case studies, in
significant benefits and a large improvement in performance when
complement these analyses with further studies on the scalability
rate on its effectiveness. We also propose a method to identify opti
cases to run and fault detection.

Categories and Subject Descriptors: D.2.4 **[Software Engineering]**: Software/Prog
[Software Engineering]: Testing and Debugging

General Terms: Verification

Additional Key Words and Phrases: Test case selection, test case minimization, model-based testing, similarity function, search-based software engineering

ACM Reference Format:
Hemmati, H., Arcuri, A., and Briand, L. 2013. Achieving scalable model-based testing through test case diversity. ACM Trans. Softw. Eng. Methodol. 22, 1, Article 6 (February 2013), 42 pages.
DOI = 10.1145/2430536.2430540 http://doi.acm.org/10.1145/2430536.2430540

L-1359, Luxembourg.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2013 ACM 1049-331X/2013/02-ART6 $15.00
DOI 10.1145/2430536.2430540 http://doi.acm.org/10.1145/2430536.2430540

ACM Transactions on Software Engineering and Methodology, Vol. 22, No. 1, Article 6, Pub. date: February 2013.

Hemmati et al. TOSEM 2013

"Evolutionary Algorithm to be the most cost-effective technique on average on two industrial case studies". Hemmati et al.

1) Model Driven Testing
   - Model-based coverage
   - Abstract Test Cases

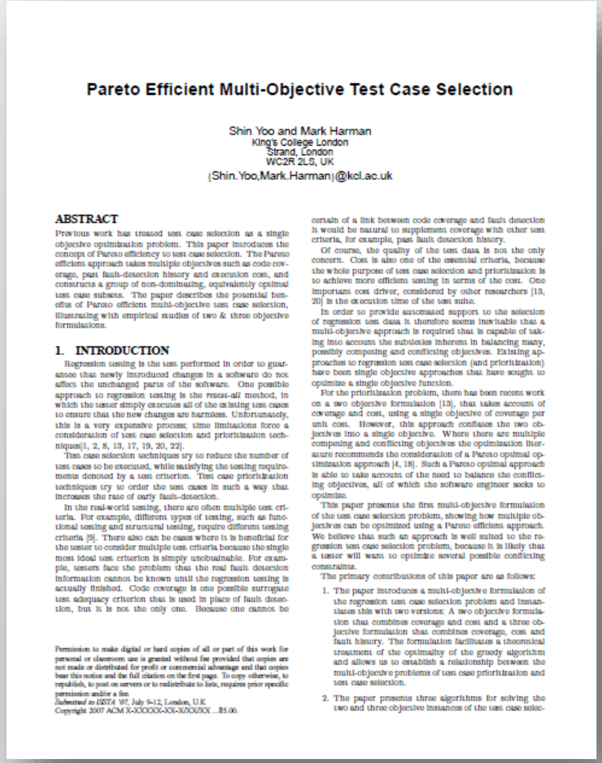
2) Search Algorithms
   - ithm
   - rch Algorithm
   - Algorithms

3) Similarity functions
   - Set distance
   - Hamming distance
   - Jaccard distance
   - Clustering distance

Multi-Criteria Regression Testing

Mono-Objective Paradigm

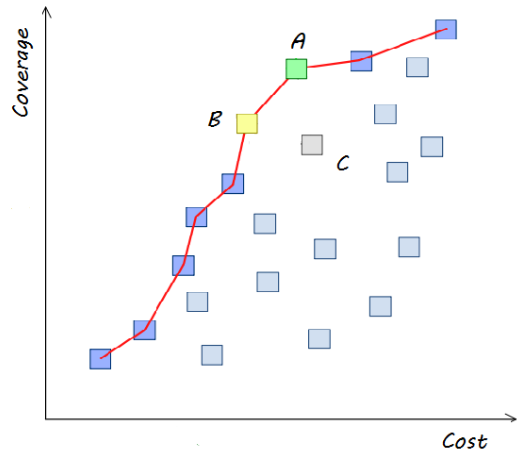
Multi-Objective Paradigm



Multi-Criteria Regression Testing

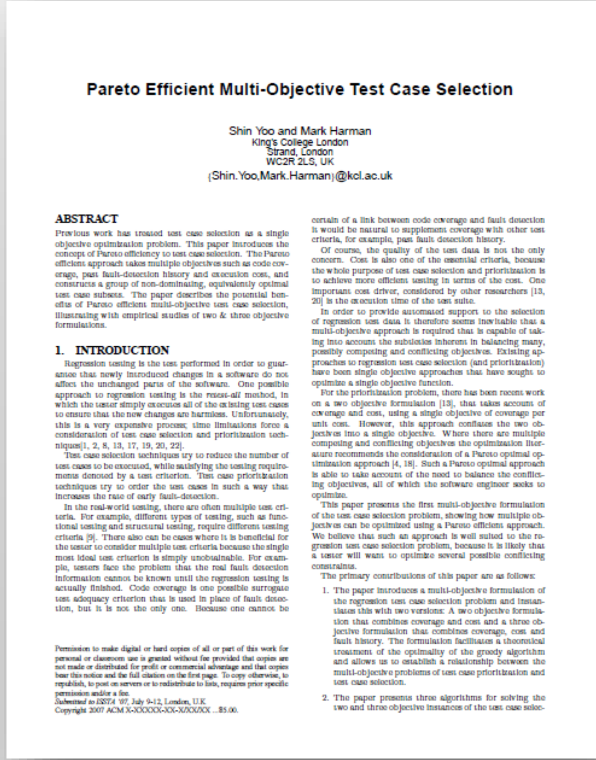
Multiple objectives are optimized using Pareto efficient approaches

Multiple otpimal solutions can be found



Pareto Optimality: all solutions that are not dominated by any other solutions form the Pareto optimal set.

Multi-Objective Paradigm



Pareto Efficient Multi-Objective Test Case Selection

Shin Yoo and Mark Harman
King's College London
Strand, London
WC2R 2LS, UK
{Shin.Yoo,Mark.Harman}@kcl.ac.uk

ABSTRACT

Previous work has treated test case selection as a single objective optimization problem. This paper introduces the concept of Pareto efficiency to test case selection. The Pareto efficient approach takes multiple objectives such as code coverage, past fault-detection history and execution cost, and constructs a group of non-dominating, equivalently optimal test case subsets. The paper describes the potential benefits of Pareto efficient multi-objective test case selection, illustrating with empirical studies of two & three objective formulations.

1. INTRODUCTION

Regression testing is the test performed in order to guarantee that newly introduced changes in a software do not affect the unchanged parts of the software. One possible approach to regression testing is the *retest-all* method, in which the tester simply executes all of the existing test cases to ensure that the new changes are harmless. Unfortunately, this is a very expensive process; time limitations force a consideration of test case selection and prioritization techniques[1, 2, 8, 13, 17, 19, 20, 22].

Test case selection techniques try to reduce the number of test cases to be executed, while satisfying the testing requirements denoted by a test criterion. Test case prioritization techniques try to order the test cases in such a way that increases the rate of early fault-detection.

In the real-world testing, there are often multiple test criteria. For example, different types of testing, such as functional testing and structural testing, require different testing criteria [9]. There also can be cases where it is beneficial for the tester to consider multiple test criteria because the single most ideal test criterion is simply unobtainable. For example, testers face the problem that the real fault detection information cannot be known until the regression testing is actually finished. Code coverage is one possible surrogate test adequacy criterion that is used in place of fault detection, but it is not the only one. Because one cannot be certain of a link between code coverage and fault detection it would be natural to supplement coverage with other test criteria, for example, past fault detection history.

Of course, the quality of the test data is not the only concern. Cost is also one of the essential criteria, because the whole purpose of test case selection and prioritization is to achieve more efficient testing in terms of the cost. One important cost driver, considered by other researchers [13, 20] is the execution time of the test suite.

In order to provide automated support to the selection of regression test data it therefore seems inevitable that a multi-objective approach is required that is capable of taking into account the subtleties inherent in balancing many, possibly competing and conflicting objectives. Existing approaches to regression test case selection (and prioritization) have been single objective approaches that have sought to optimize a single objective function.

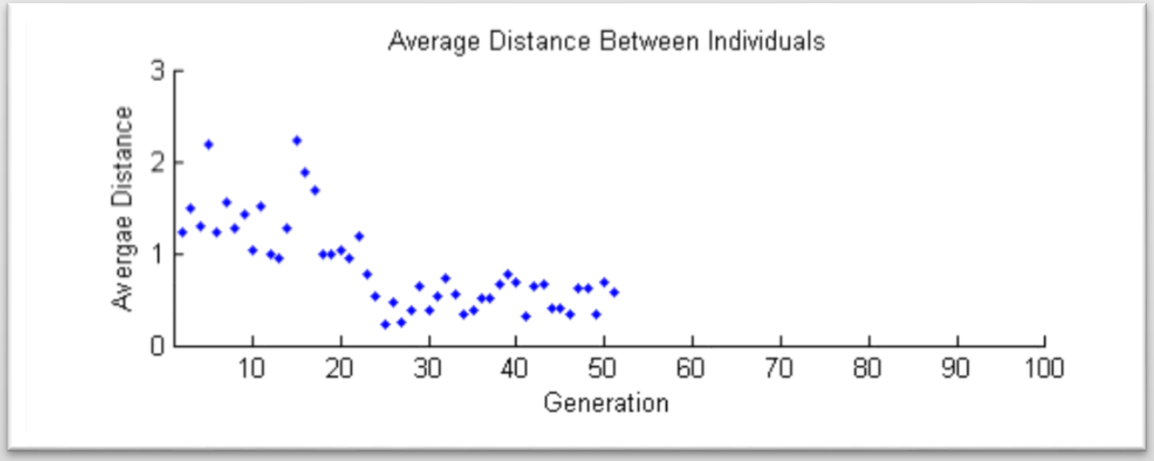
For the prioritization problem, there has been recent work on a two objective formulation [13], that takes account of coverage and cost, using a single objective of coverage per unit cost. However, this approach conflates the two objectives into a single objective. Where there are multiple competing and conflicting objectives the optimization literature recommends the consideration of a Pareto optimal optimization approach [4, 18]. Such a Pareto optimal approach is able to take account of the need to balance the conflicting objectives, all of which the software engineer seeks to optimize.

This paper presents the first multi-objective formulation of the test case selection problem, showing how multiple objectives can be optimized using a Pareto efficient approach. We believe that such an approach is well suited to the regression test case selection problem, because it is likely that a tester will want to optimize several possible conflicting constraints.

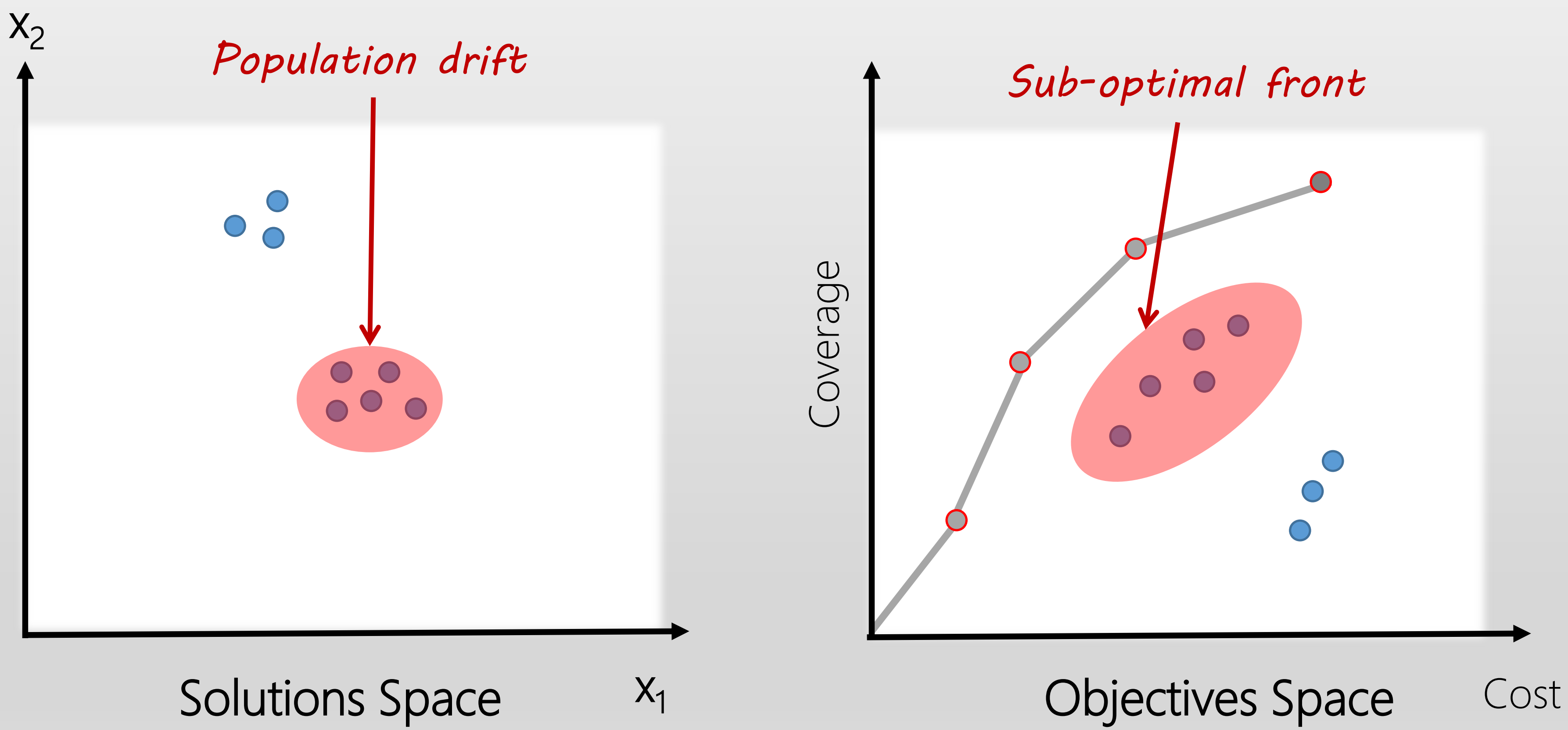
The primary contributions of this paper are as follows:

1. The paper introduces a multi-objective formulation of the regression test case selection problem and instantiates this with two versions: A two objective formulation that combines coverage and cost and a three objective formulation that combines coverage, cost and fault history. The formulation facilitates a theoretical treatment of the optimality of the greedy algorithm and allows us to establish a relationship between the multi-objective problems of test case prioritization and test case selection.
2. The paper presents three algorithms for solving the two and three objective instances of the test case selec-

Diversity in Evolutionary Algorithms

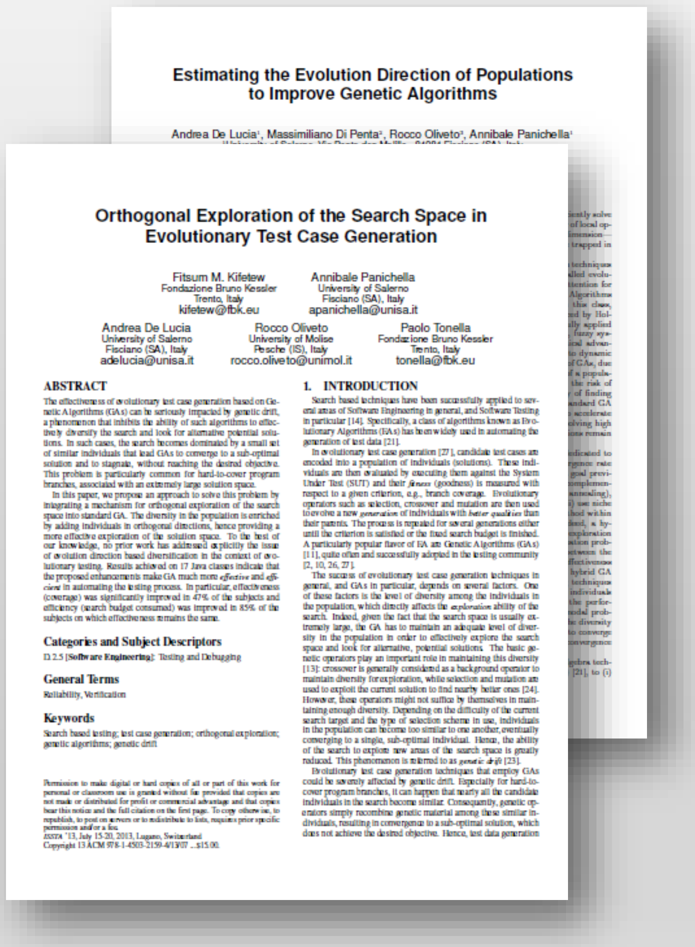


Diversity in Evolutionary Algorithms



Injecting Diversity during the Evolution

Orthogonal exploration



Estimating the Evolution Direction of Populations to Improve Genetic Algorithms

Andrea De Lucia[1], Massimiliano Di Penta[2], Rocco Oliveto[3], Annibale Panichella[1]

Orthogonal Exploration of the Search Space in Evolutionary Test Case Generation

Fitsum M. Kifetew
Fondazione Bruno Kessler
Trento, Italy
kifetew@fbk.eu

Annibale Panichella
University of Salerno
Fisciano (SA), Italy
apanichella@unisa.it

Andrea De Lucia
University of Salerno
Fisciano (SA), Italy
adelucia@unisa.it

Rocco Oliveto
University of Molise
Pesche (IS), Italy
rocco.oliveto@unimol.it

Paolo Tonella
Fondazione Bruno Kessler
Trento, Italy
tonella@fbk.eu

ABSTRACT

Categories and Subject Descriptors

General Terms

Keywords

1. INTRODUCTION

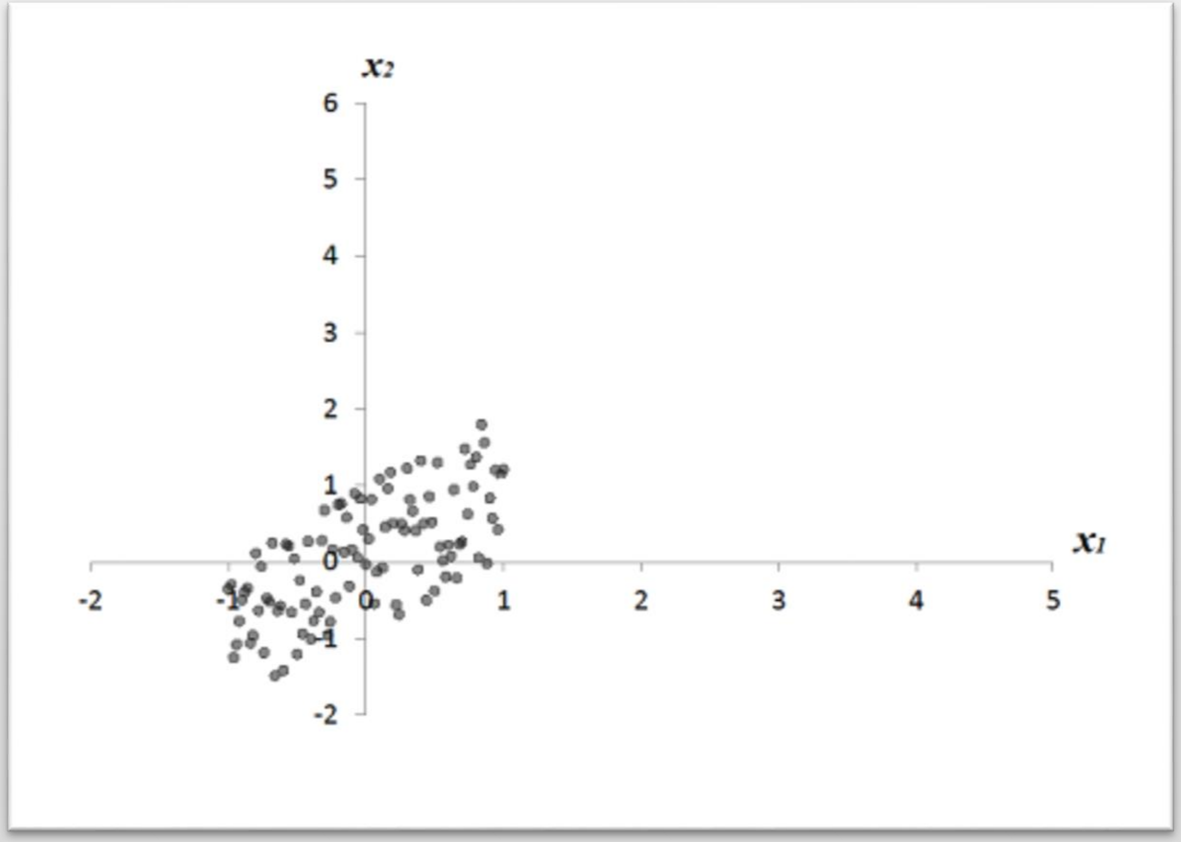
Estimating the Evolution Direction of Populations to Improve Genetic Algorithm. A. De Lucia , M. Di Penta, R. Oliveto, **A. Panichella**

GECCO 2012

Orthogonal Exploration of the Search Space in Evolutionary Test Case Generation F. M. Kifetew, **A. Panichella** , A. De Lucia , R. Oliveto, P. Tonella

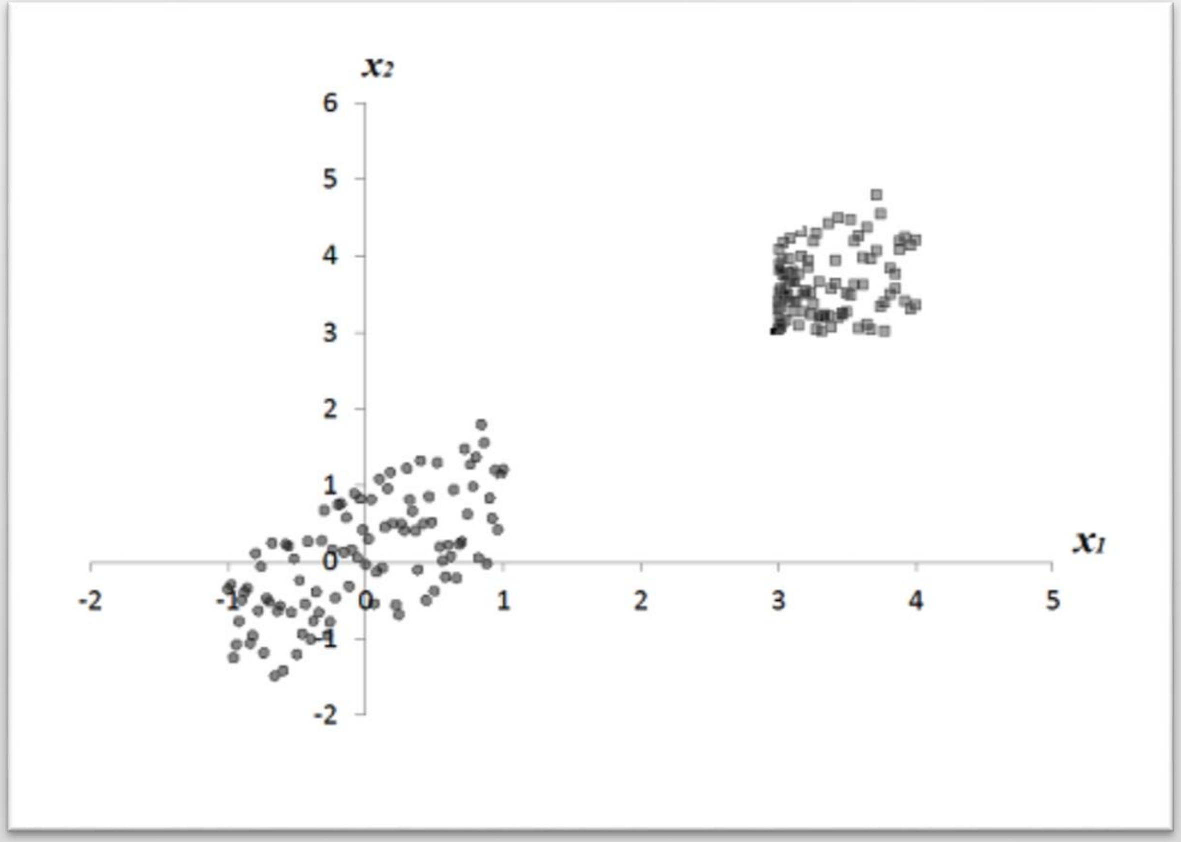
ISSTA 2013

What is the evolution direction?



P(t) = Population at generation t

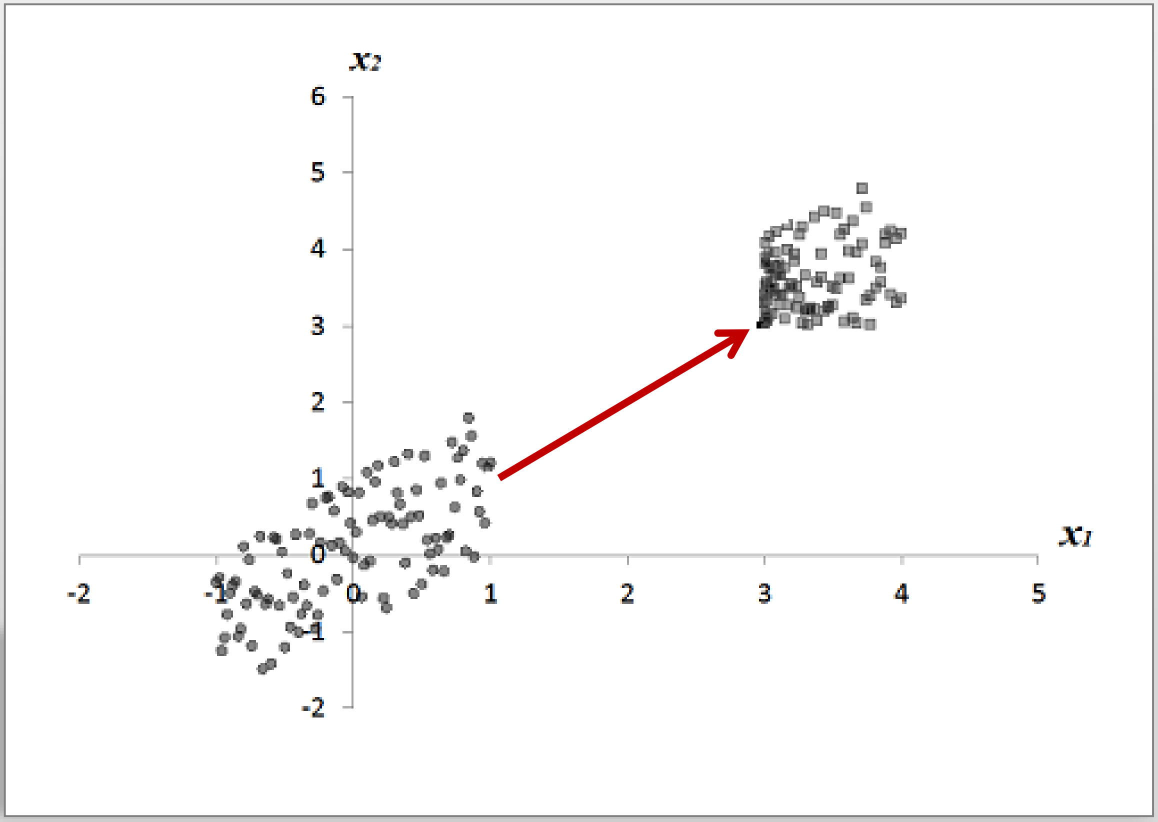
What is the evolution direction?



P(t) = Population at generation t

P(t+k) = Population after k generations

What is the evolution direction?

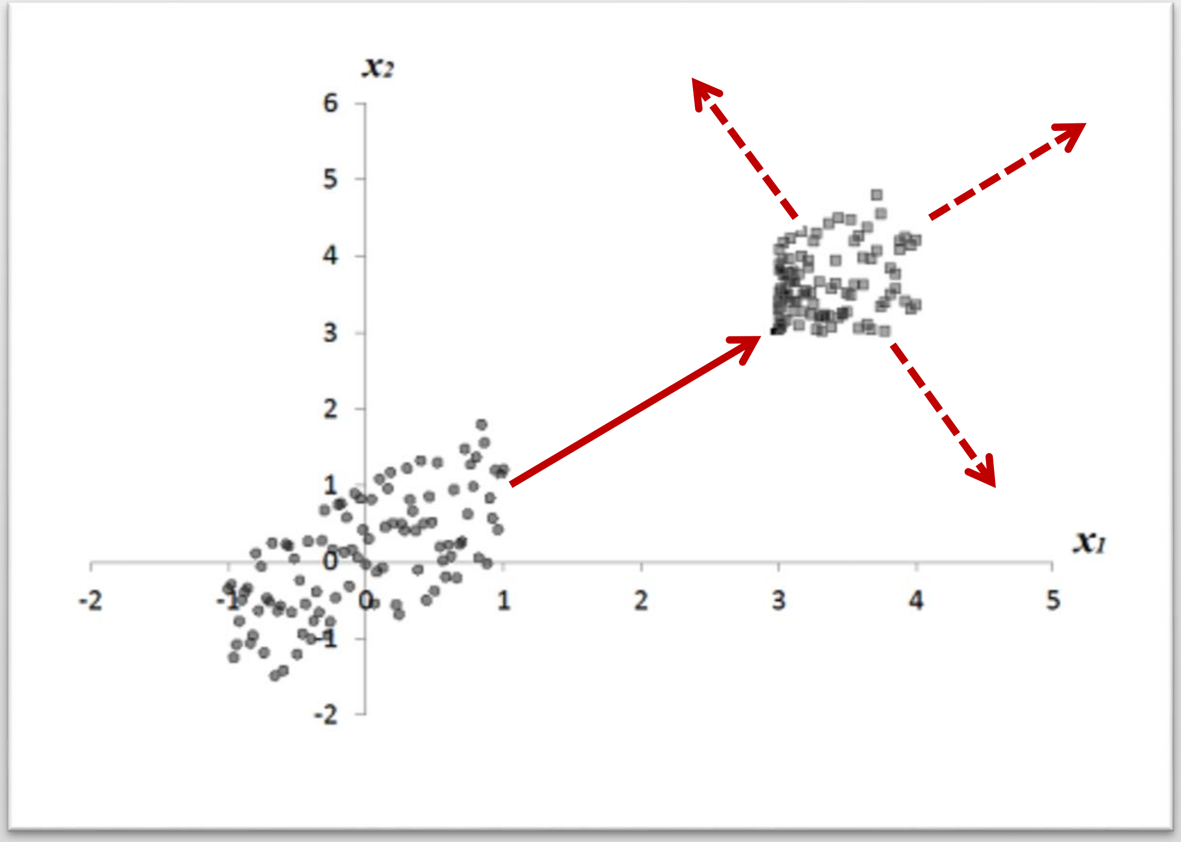


P(t) = Population at generation t

P(t+k) = Population after k generations

Evolution Directions

Why?



P(t) = Population at generation t

P(t+k) = Population after k generations

Evolution Directions

Orthogonal Individuals

How?

The basic idea is that a population of solutions P provided by GA at generation t can be viewed as a m x n matrix

variable 1 variable 2 variable n

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,n} \end{bmatrix}$$

Individual 1
Individual 2
Individual m

How? Singual Value Decomposition

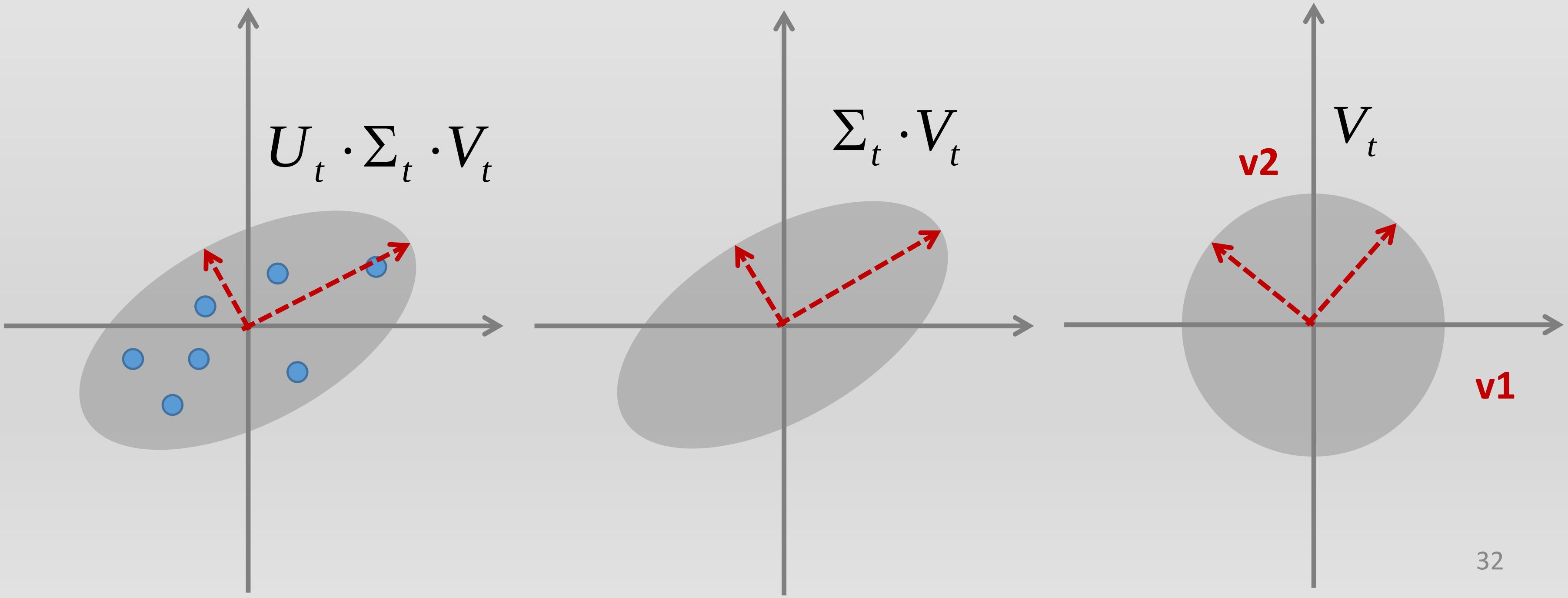
Each matrix P can be decomposed in the product of three different matrices:

$$\underset{m \times n}{P} = \underset{m \times k}{U} \cdot \underset{k \times k}{\Sigma} \cdot \underset{k \times n}{V}$$

How? Singual Value Decomposition

Each matrix P can be decomposed in the product of three different matrices:

$$\underset{m\times n}{P} = \underset{m\times k}{U} \cdot \underset{k\times k}{\Sigma} \cdot \underset{k\times n}{V}$$



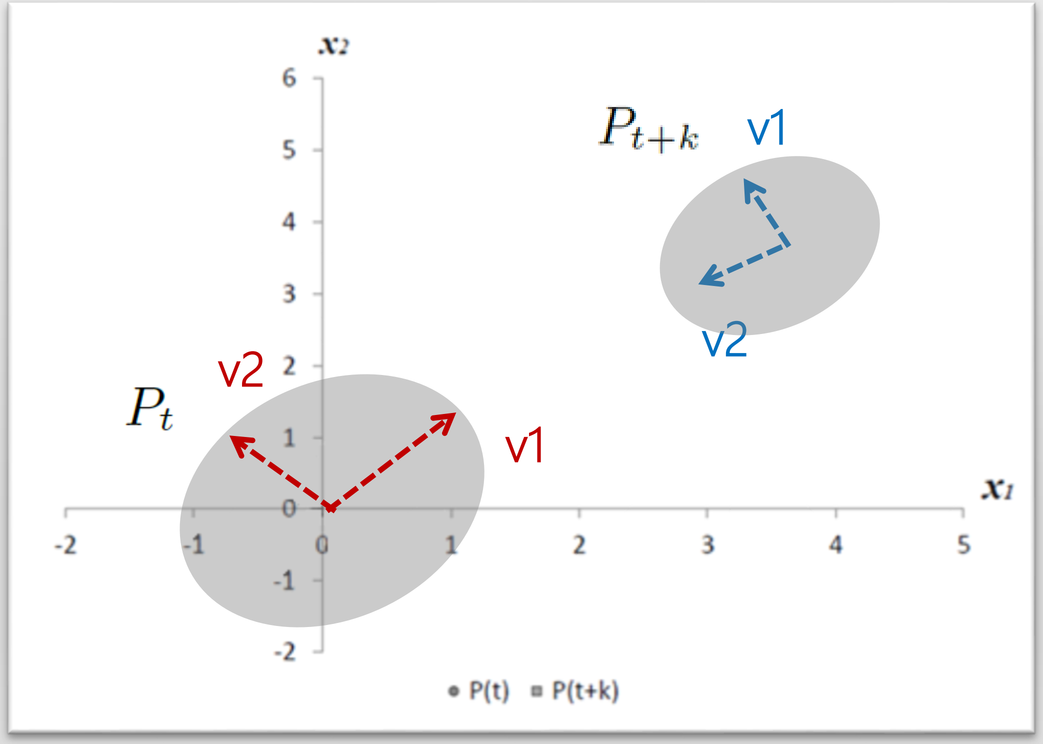
Using SVD for Evolution Direction

Population at generation t

$$P_t = U_t \cdot \Sigma_t \cdot V_t$$

Population at generation t + k

$$P_{t+k} = U_{t+k} \cdot \Sigma_{t+k} \cdot V_{t+k}$$



The currect evolution direction is related to

$$\overline{V} = V_{t+k} - V_t$$

$$\overline{\Sigma} = \Sigma_{t+k} - \Sigma_t$$

Using SVD for Evolution Direction

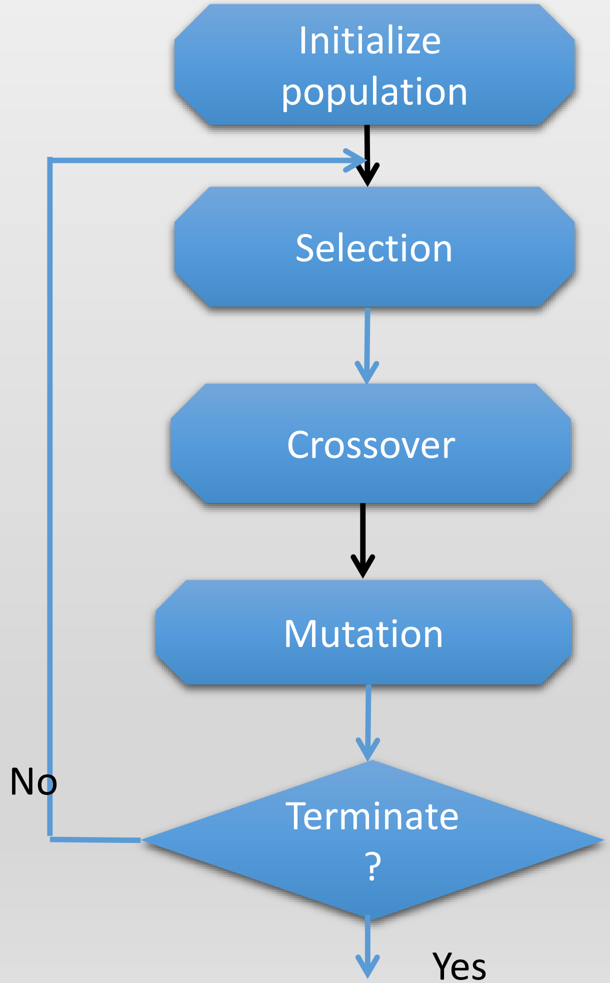
Generating new orthogonal individuals through the following equation:

$$P_{t+k}^{*} = U_{t+k} \cdot \left(\Sigma_{t+k} + \overline{\Sigma}\right) \cdot \left(V_{t+k} + \overline{V_{\perp}}\right)$$

Shifting Operator:
P_{t+k}^{*} is $\bar{\Sigma}$ shifted in the search space

Rotating Operator:
P_{t+k}^{*} is rotated with respect to P_{t+k}

Integration SVD with NSGA-II

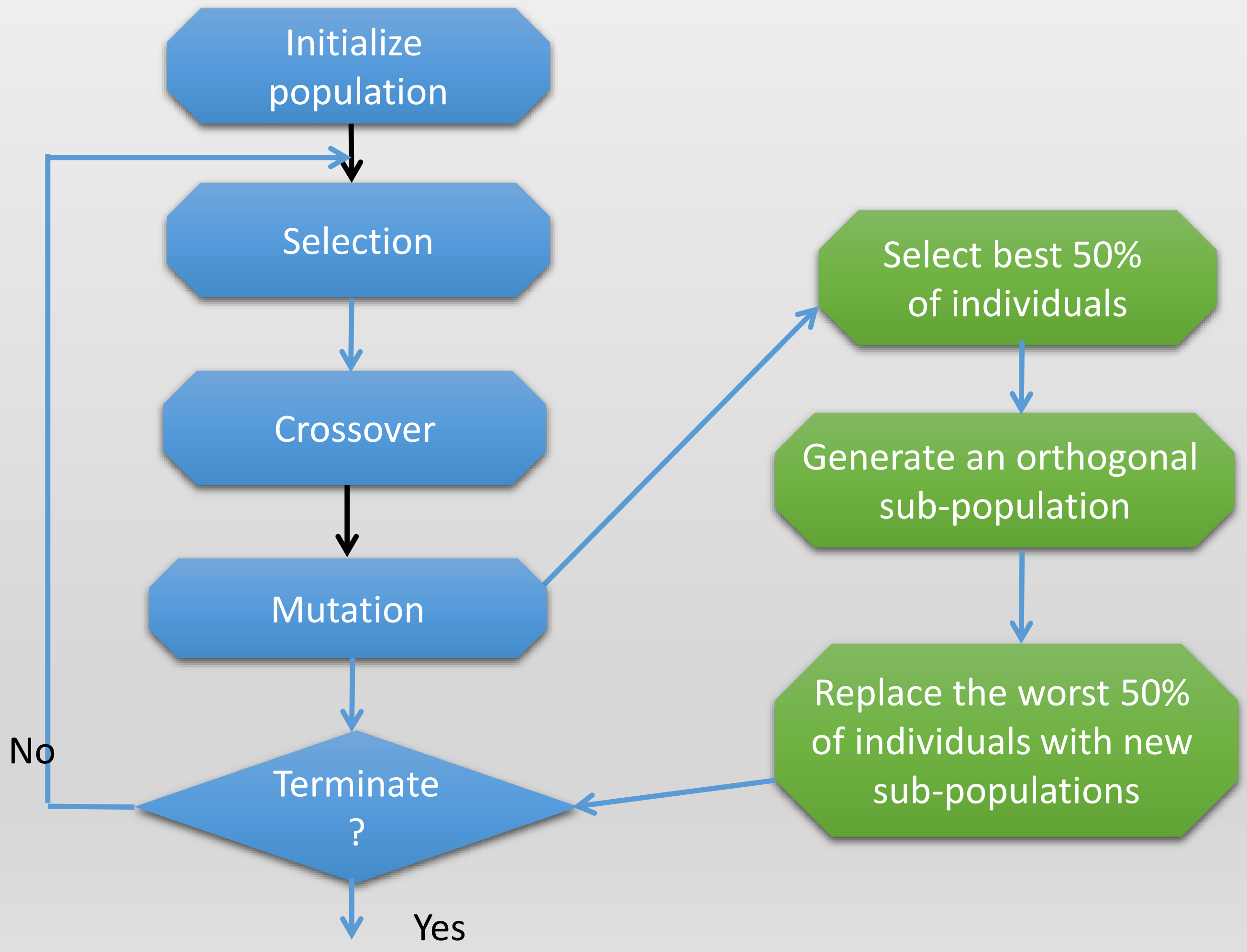


- Non Dominated Sorting Algorithm
- Crowding Distance
- Tournament Selection

- Multi-points crossover

- Bit-flip mutation

SVD + NSGA-II


Initialize population
Selection
Crossover
Mutation
Terminate ?
No
Yes
Select best 50% of individuals
Generate an orthogonal sub-population
Replace the worst 50% of individuals with new sub-populations

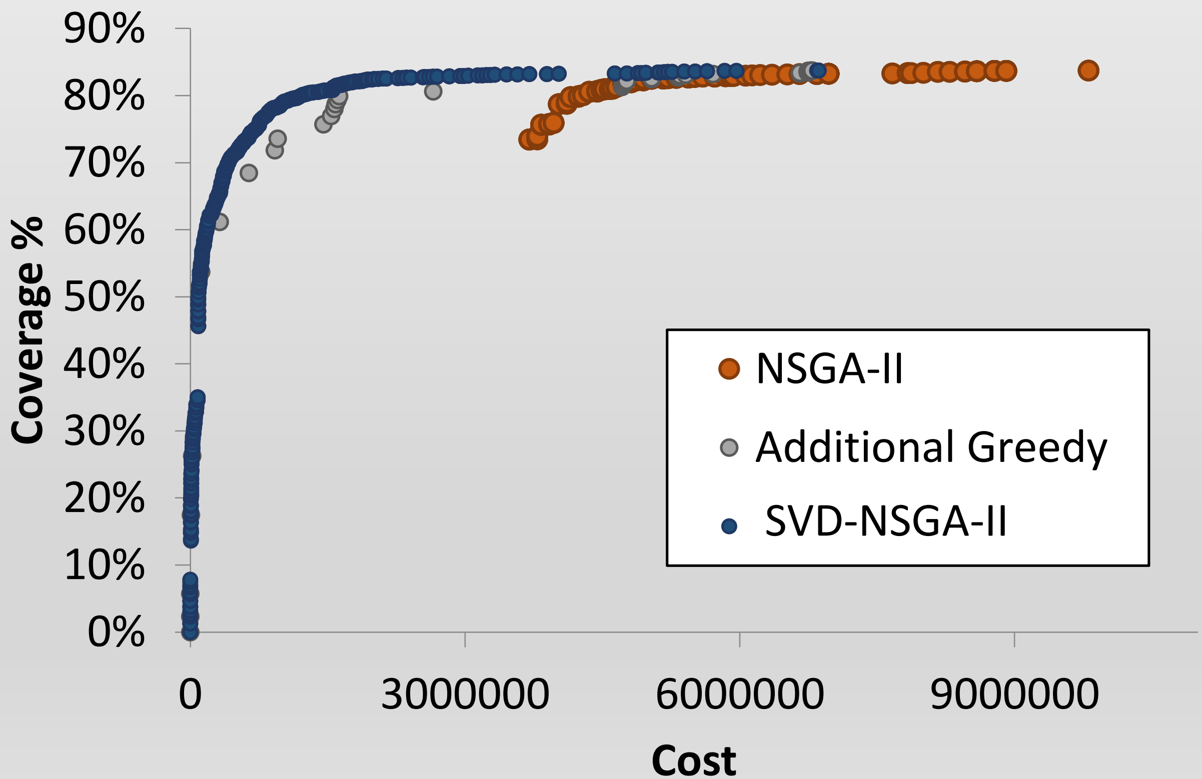
Empirical Evaluation

Case Study Design

RQs	Systems	Algorithms	Metrics
RQ1: To what extent does SVD-NSGA-II produce near optimal solutions, compared to alternative techniques?	Bash, Flex, Grep, Gzip, Printtokens, Printtokens2 Schedule, Schedule2, Sed, Space, vim	Add.Greedy Algorithm NSGA-II SVD-NSGA-II	1. Pareto Front Size 2. N. non-dominated solutions
RQ2: What is the cost-effectiveness of SVD-NSGA-II compared to the alternative techniques?	Bash, Flex, Grep, Gzip, Printtokens, Printtokens2 Schedule, Schedule2, Sed, Space, vim	Add.Greedy Algorithm NSGA-II SVD-NSGA-II	1. Fault Detection Rate

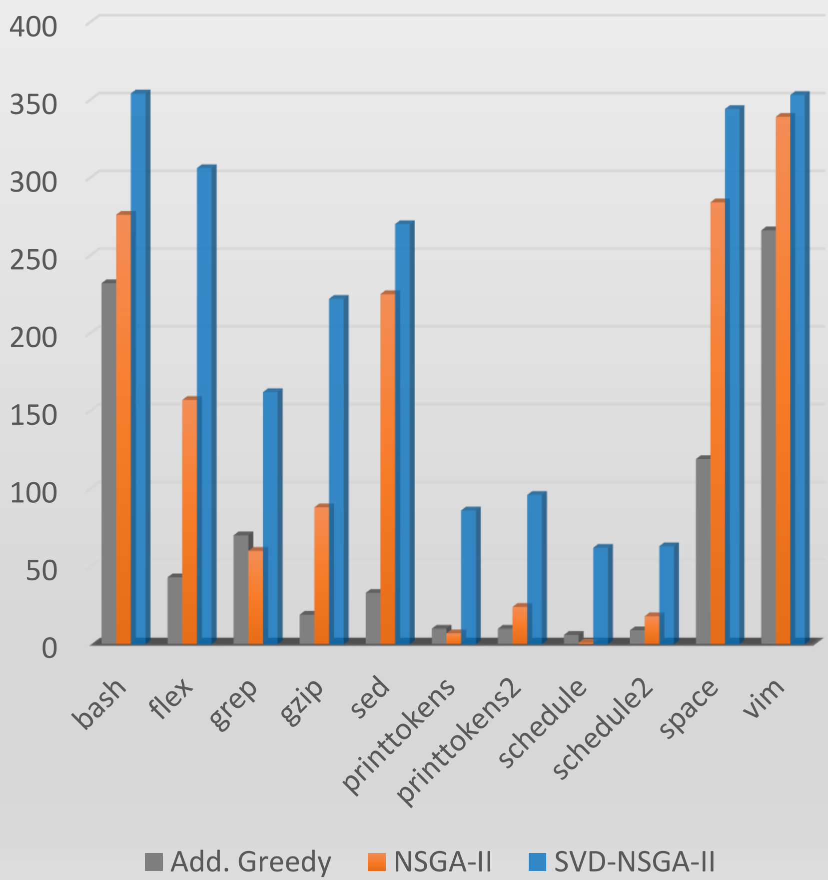
Results

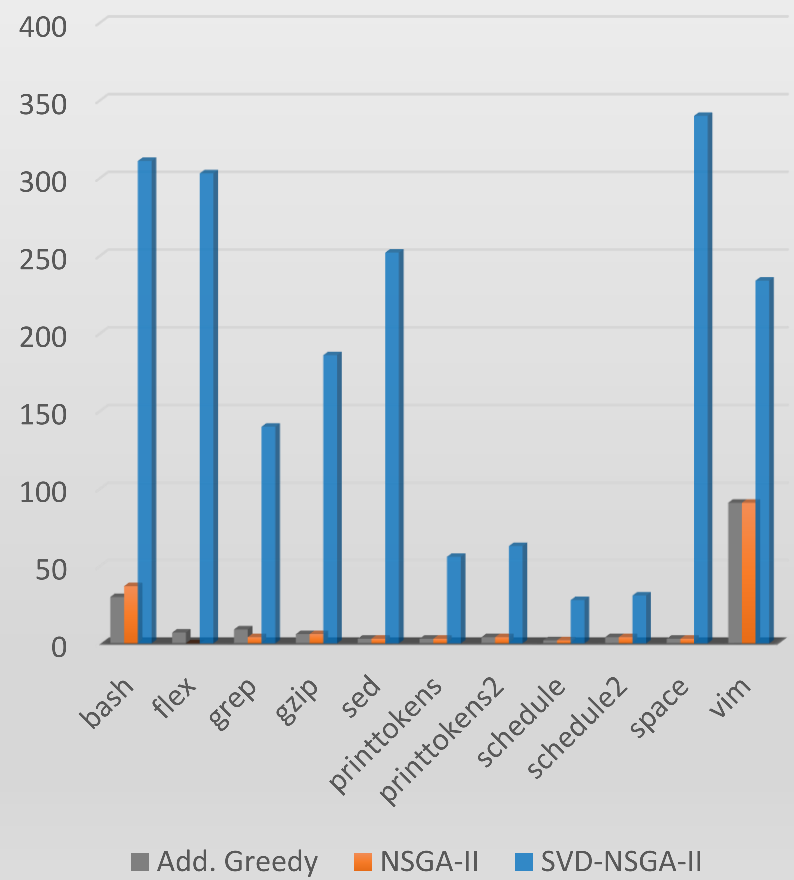
RQ1: To what extent does SVD-NSGA-II produce near optimal solutions, compared to alternative techniques?



Results

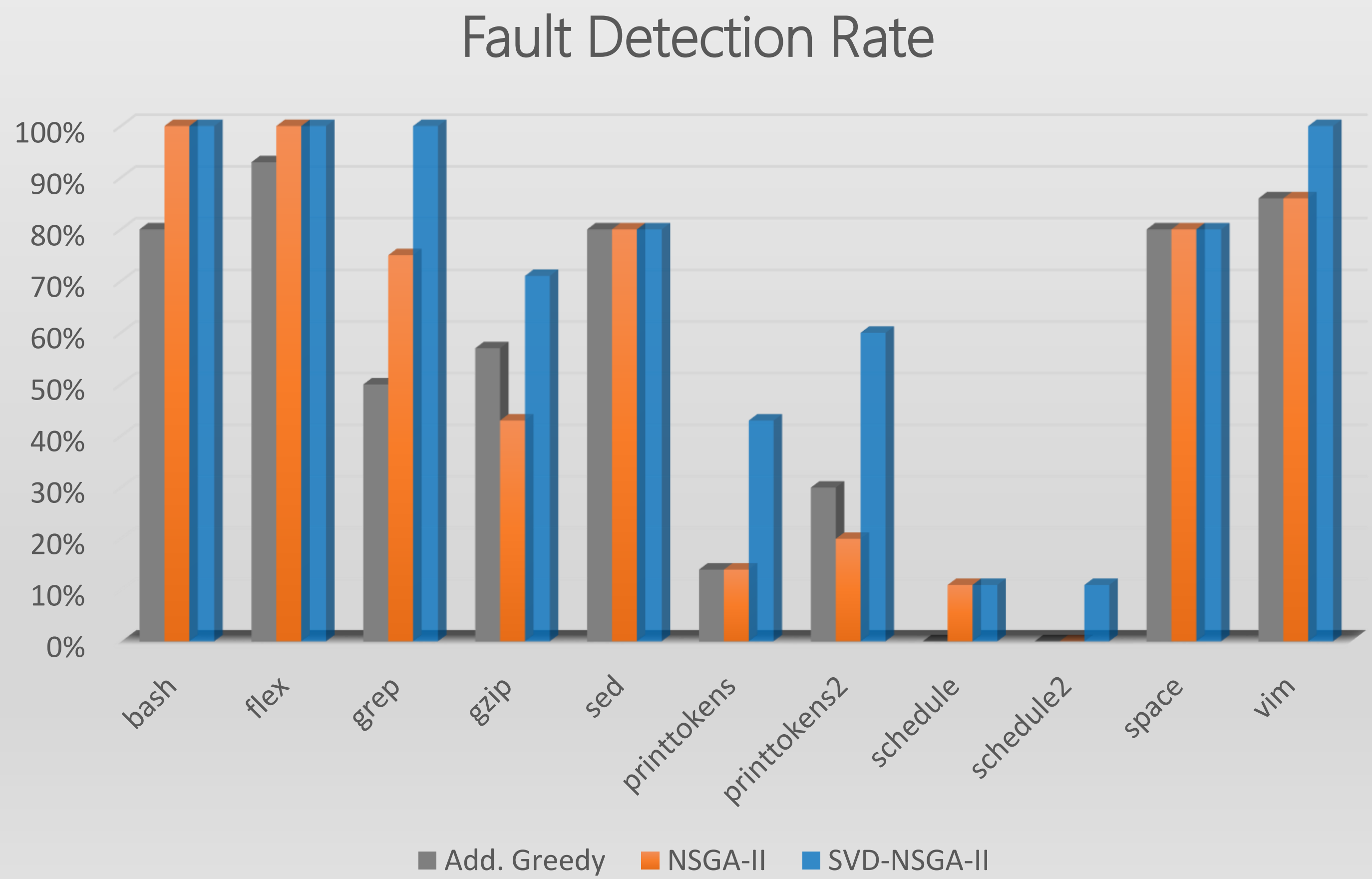
RQ1: To what extent does SVD-NSGA-II produce near optimal solutions, compared to alternative techniques?





Results

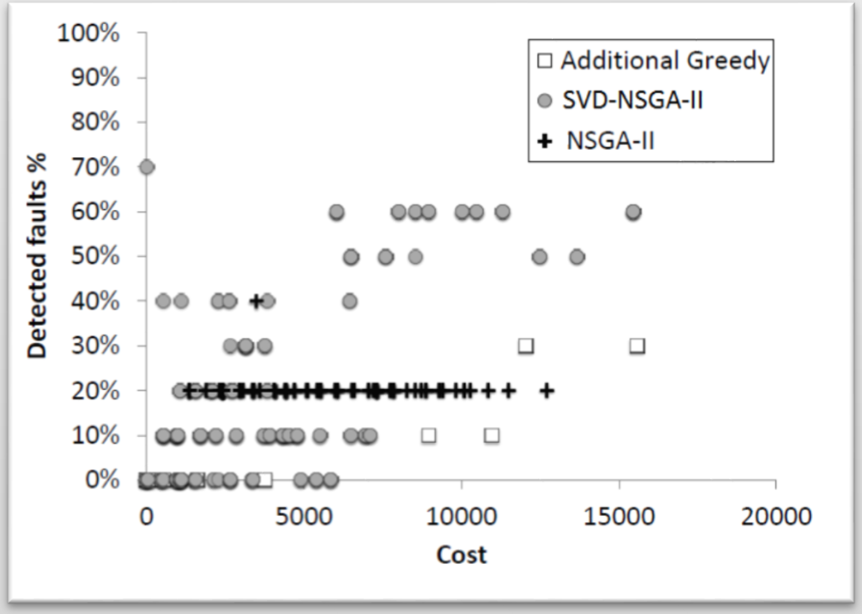
RQ2: What is the cost-effectiveness of SVD-NSGA-II compared to the alternative techniques?



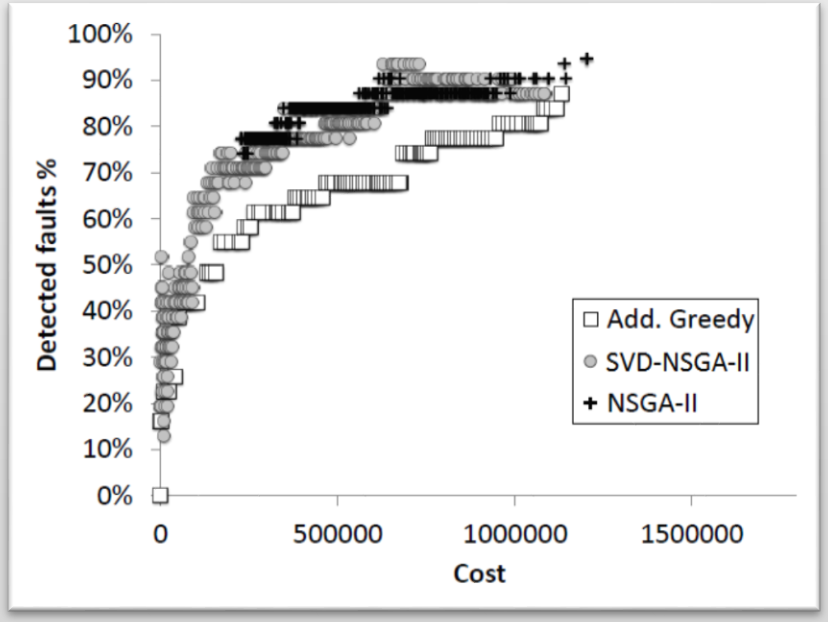
Results

RQ2: What is the cost-effectiveness of SVD-NSGA-II compared to the alternative techniques?

printtokens2



space

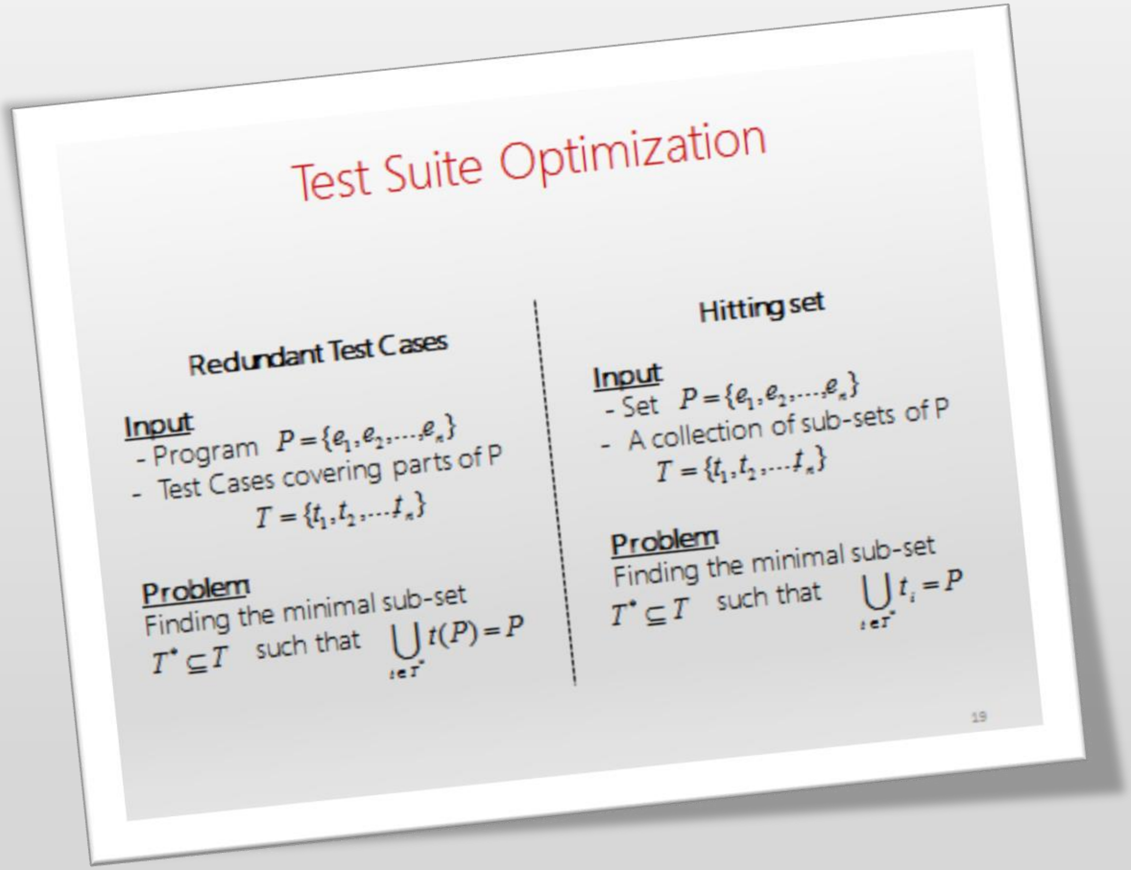


Summary

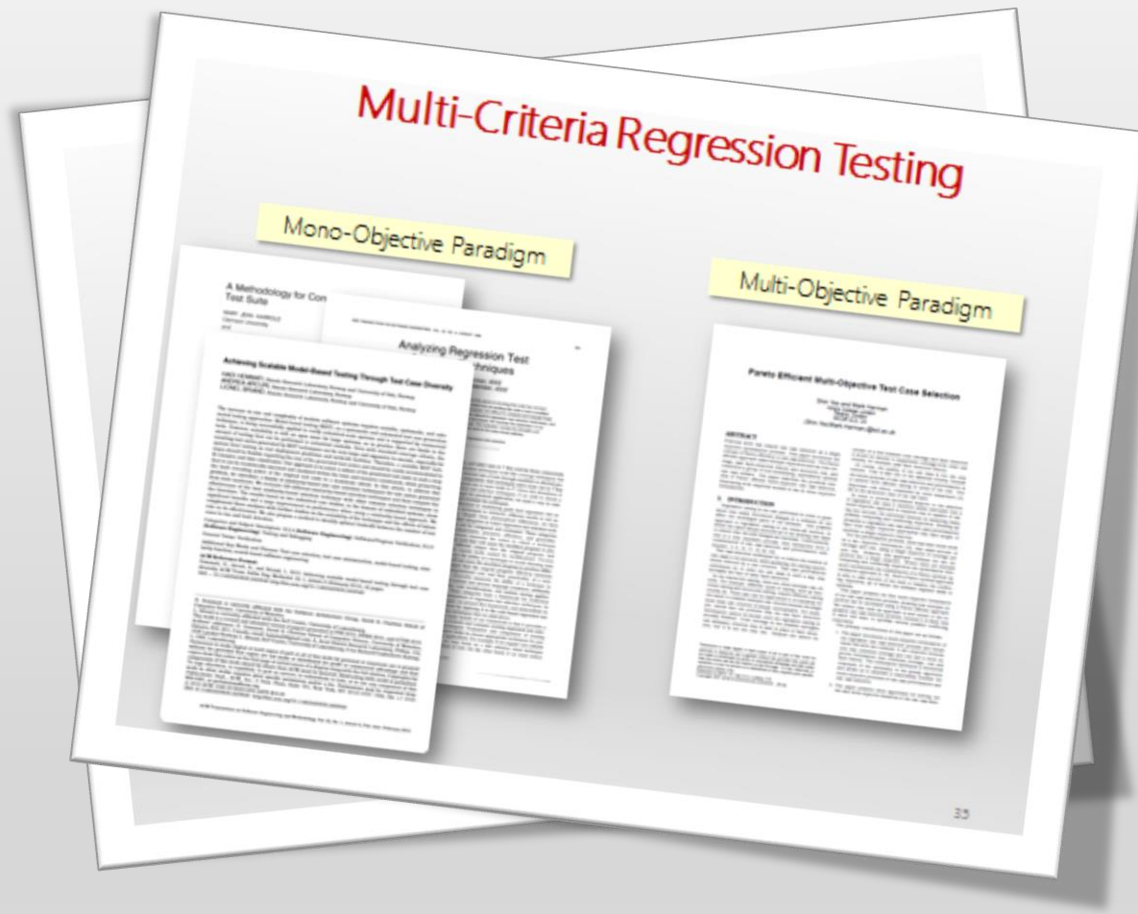
100% The optimality was improved

The effectiveness was improved at same level of execution cost 64%

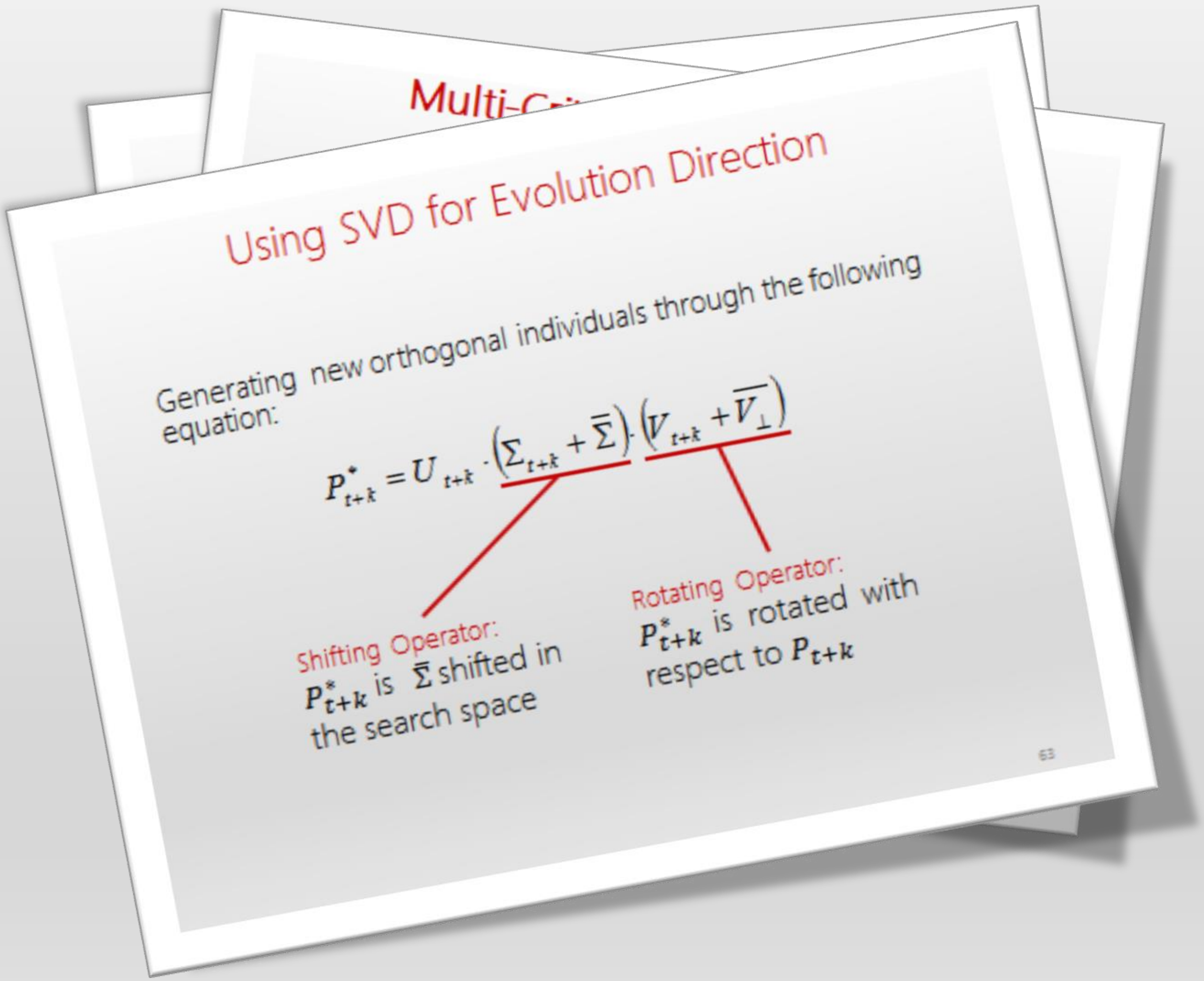
Conclusion


Test Suite Optimization
Redundant Test Cases
Input
- Program $P=\{e_1,e_2,\ldots,e_n\}$
- Test Cases covering parts of P
$T=\{t_1,t_2,\ldots t_n\}$
Problem
Finding the minimal sub-set
$T^{*}\subseteq T$ such that $\bigcup_{t\in T^{*}} t(P)=P$
Hitting set
Input
- Set $P=\{e_1,e_2,\ldots,e_n\}$
- A collection of sub-sets of P
$T=\{t_1,t_2,\ldots t_n\}$
Problem
Finding the minimal sub-set
$T^{*}\subseteq T$ such that $\bigcup_{t\in T^{*}} t_i=P$
19

Conclusion


Multi-Criteria Regression Testing
Mono-Objective Paradigm
Multi-Objective Paradigm
Achieving Scalable Model-Based Testing Through Test Case Diversity
Pareto Efficient Multi-Objective Test Case Selection
35

Conclusion


Multi-Cri
Using SVD for Evolution Direction
Generating new orthogonal individuals through the following equation:
$P^*_{t+k} = U_{t+k} \cdot (\Sigma_{t+k} + \bar{\Sigma}) \cdot (V_{t+k} + \overline{V_\perp})$
Shifting Operator:
P^*_{t+k} is $\bar{\Sigma}$ shifted in the search space
Rotating Operator:
P^*_{t+k} is rotated with respect to P_{t+k}
63

Conclusion


Summary
100%
The optimality was improved
The effectiveness was improved
at same level of execution cost
64%


Thank You