Search-Based Software Maintenance and Evolution

Annibale Panichella

Advisor:
Prof. Andrea De Lucia
Dott. Rocco Oliveto

Search-Based Software Engineering

«The application of meta-heuristic search-based optimization techniques to find near-optimal solutions in software engineering problems.»

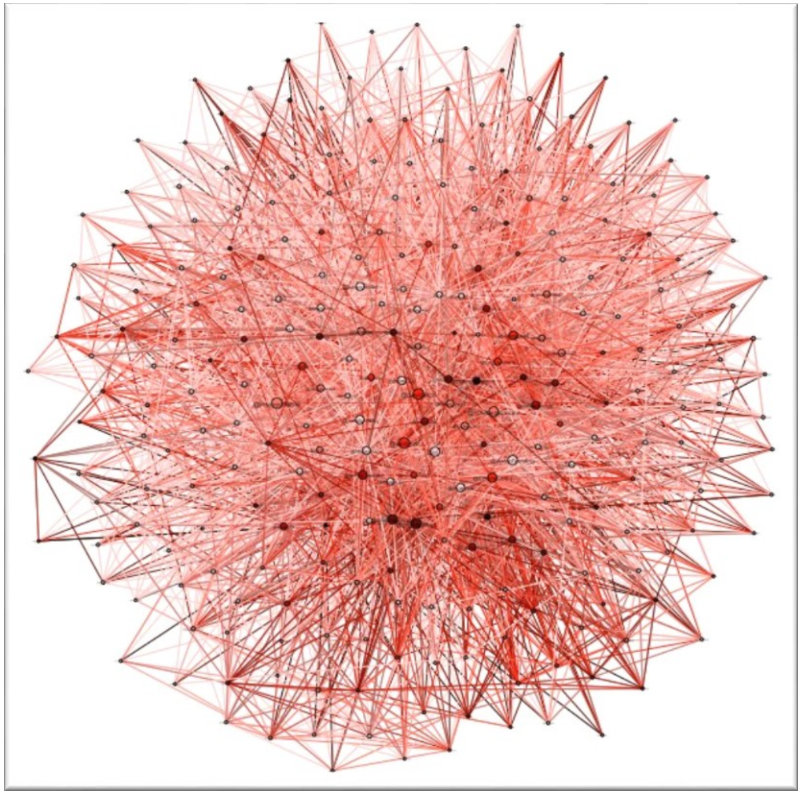
1. **Problem Reformulation**: reformulating typical SE problems as optimization problems

2. **Fitness Function**: definition of functions to optimize

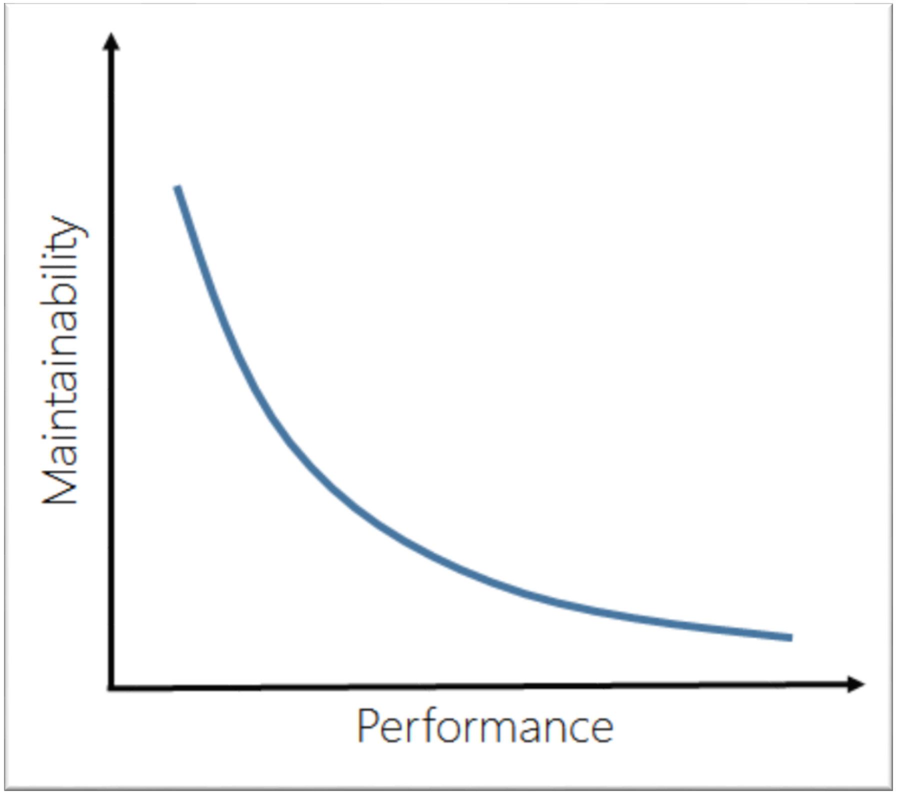
3. **Optimization Algorithms**: applying search algorithm to solve such functions
   - Genetic Algorithms
   - Hill climbing
   - Simulated Annealing
   - Random Search
   - Tabu Search
   - Particle Swarm Optimization
   - …

Why SBSE?

Large Search Space

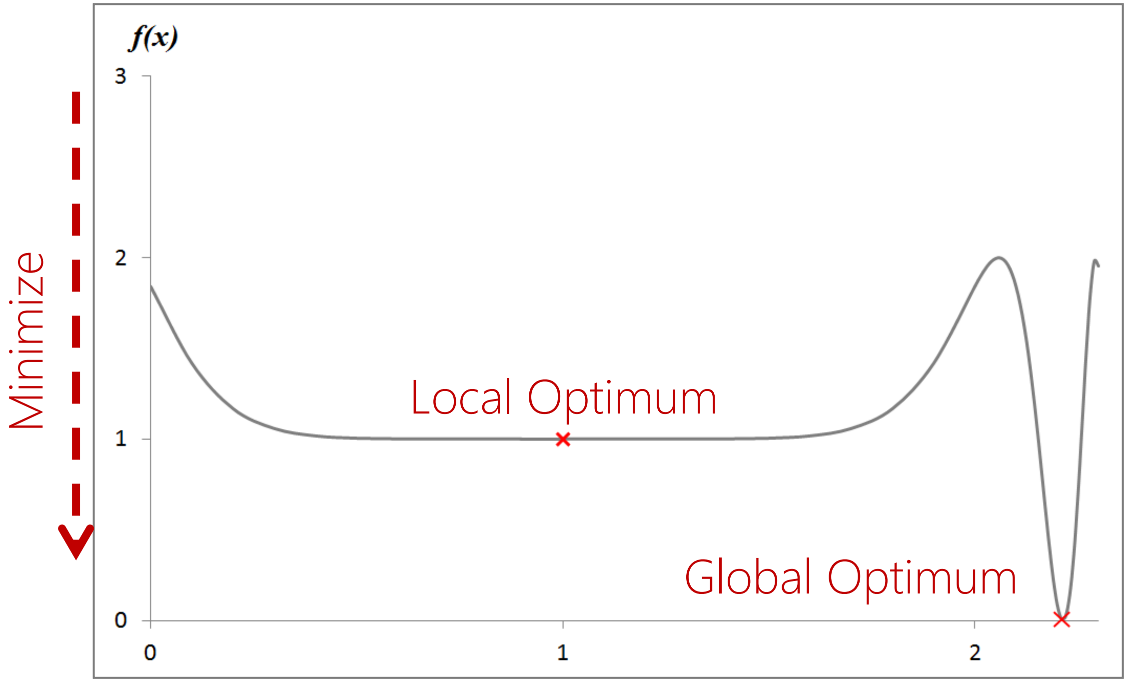


Presence of conflicting goals



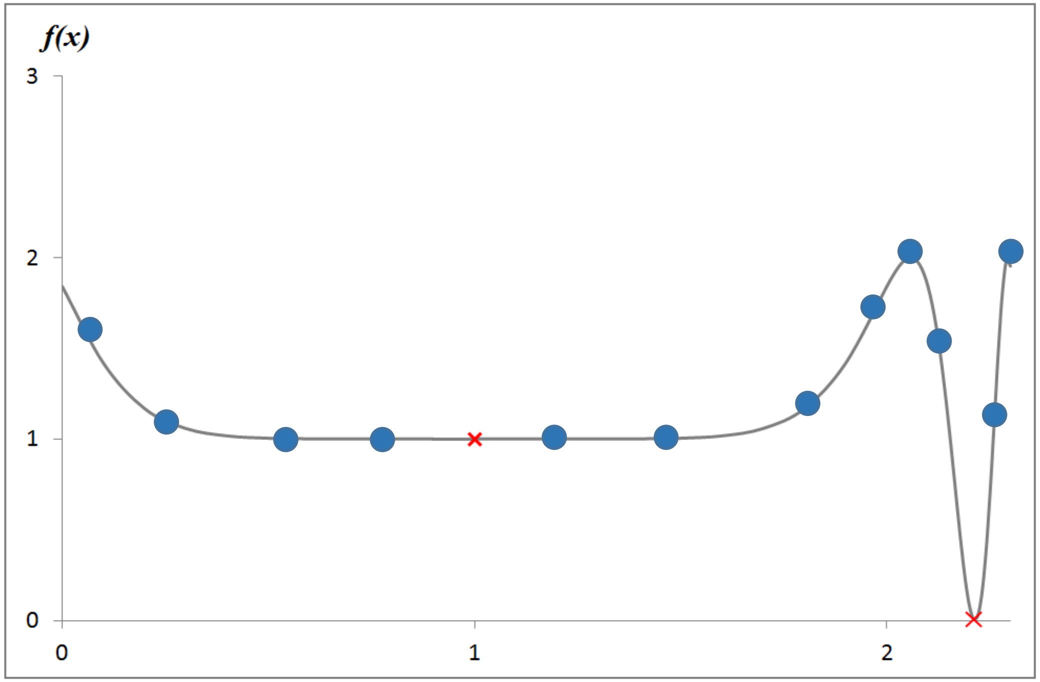
Optimization Problem

$$\min \; f(x) = \sin\left((x-1)^8\right) + 1$$



Genetic Algorithms (GAs)

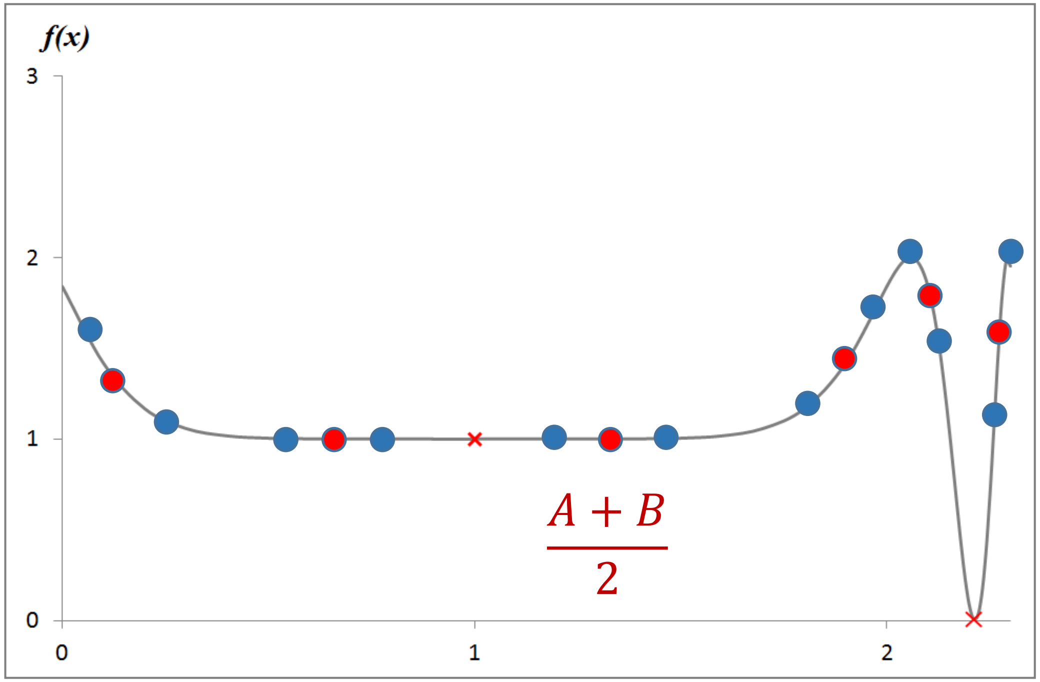
$$\min\ f(x) = \sin\left((x-1)^8\right) + 1$$

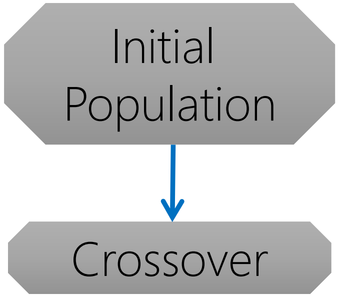


Initial Population

Genetic Algorithms (GAs)

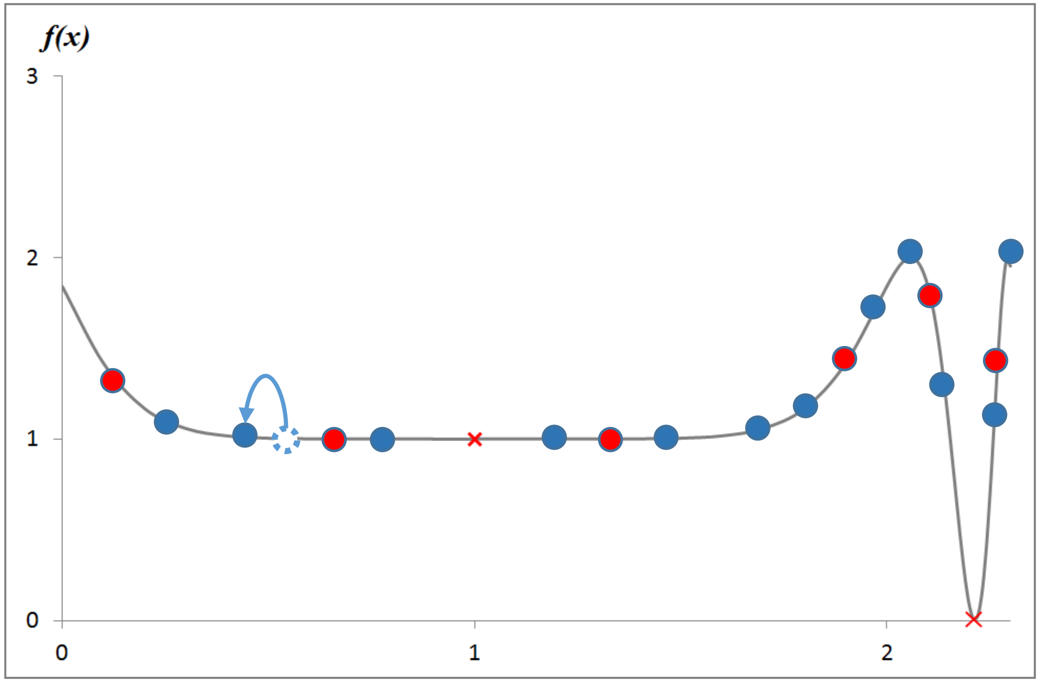
$$\min\ f(x) = \sin\left((x-1)^8\right) + 1$$

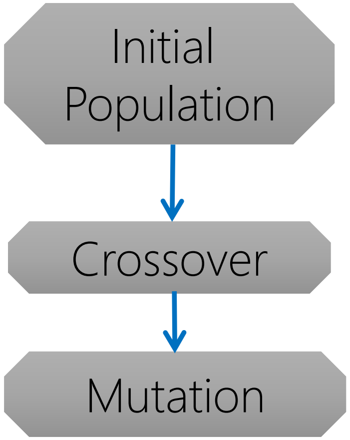




Genetic Algorithms (GAs)

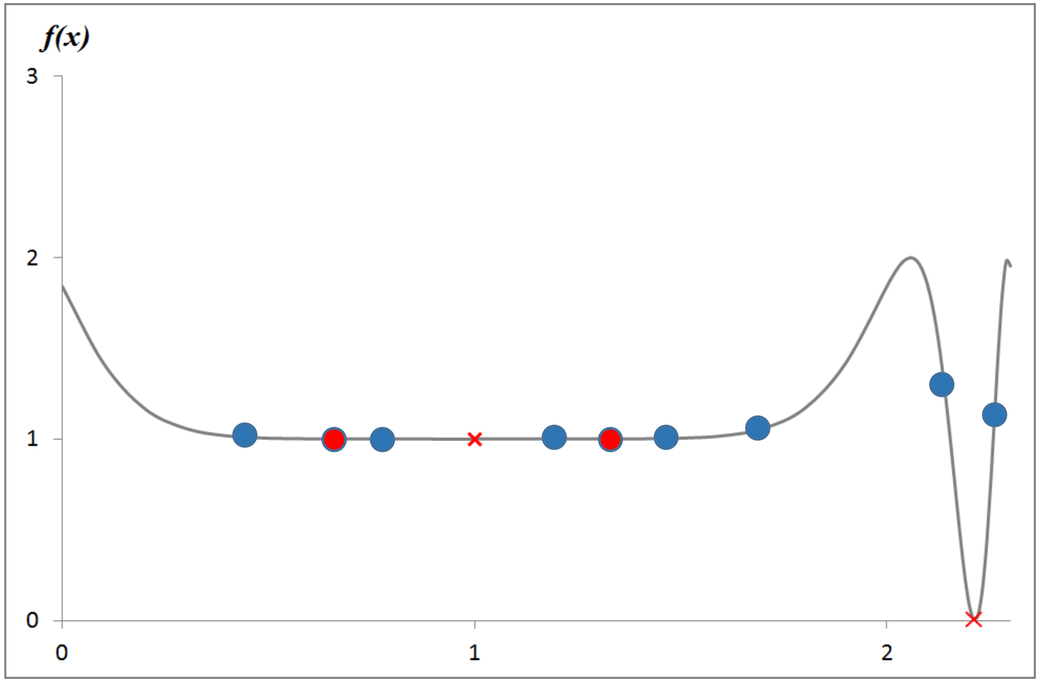
$$\min\ f(x) = \sin\left((x-1)^8\right) + 1$$

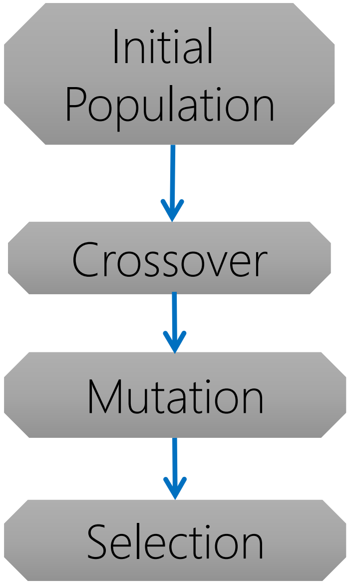




Genetic Algorithms (GAs)

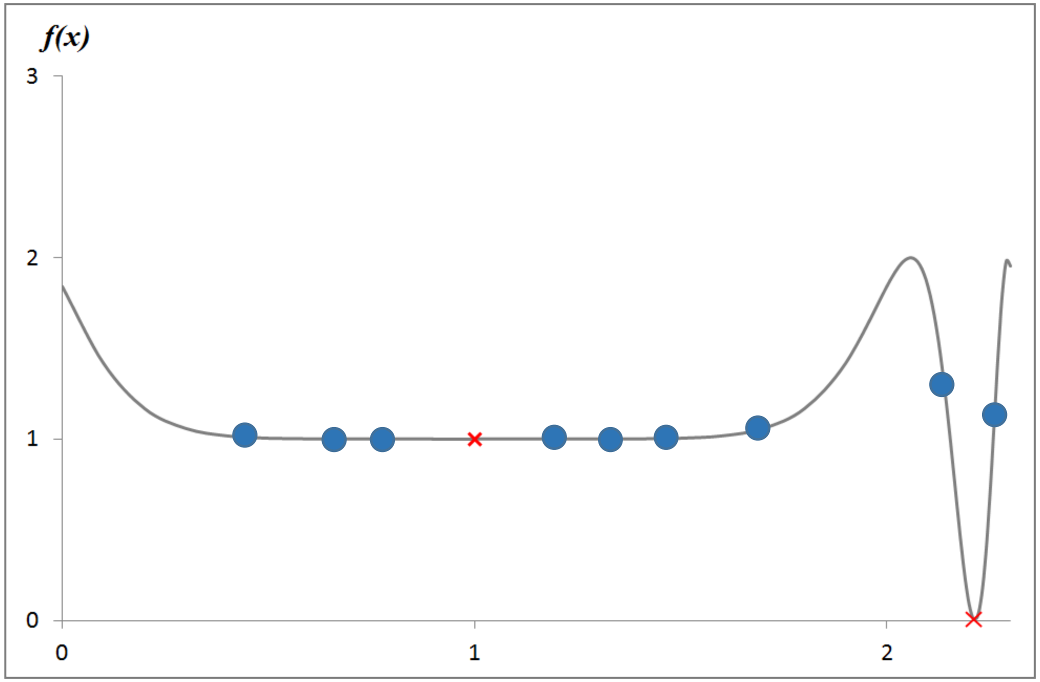
$$\min \ f(x) = \sin\left((x-1)^8\right) + 1$$

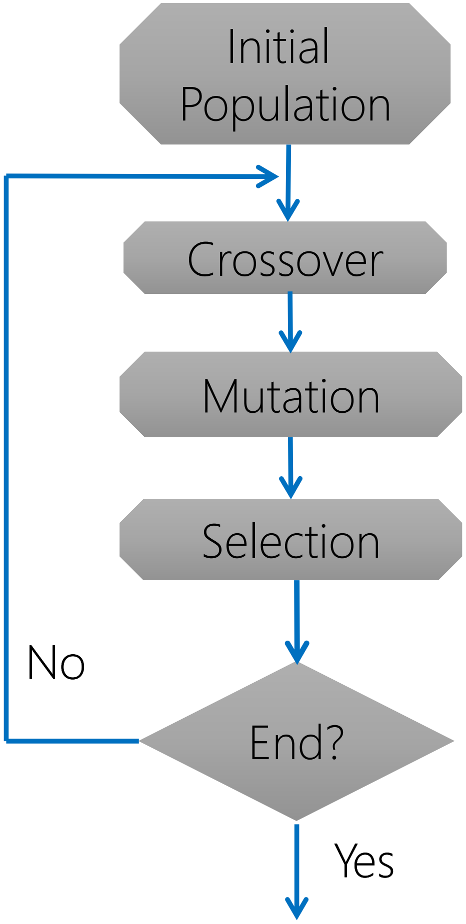




Genetic Algorithms (GAs)

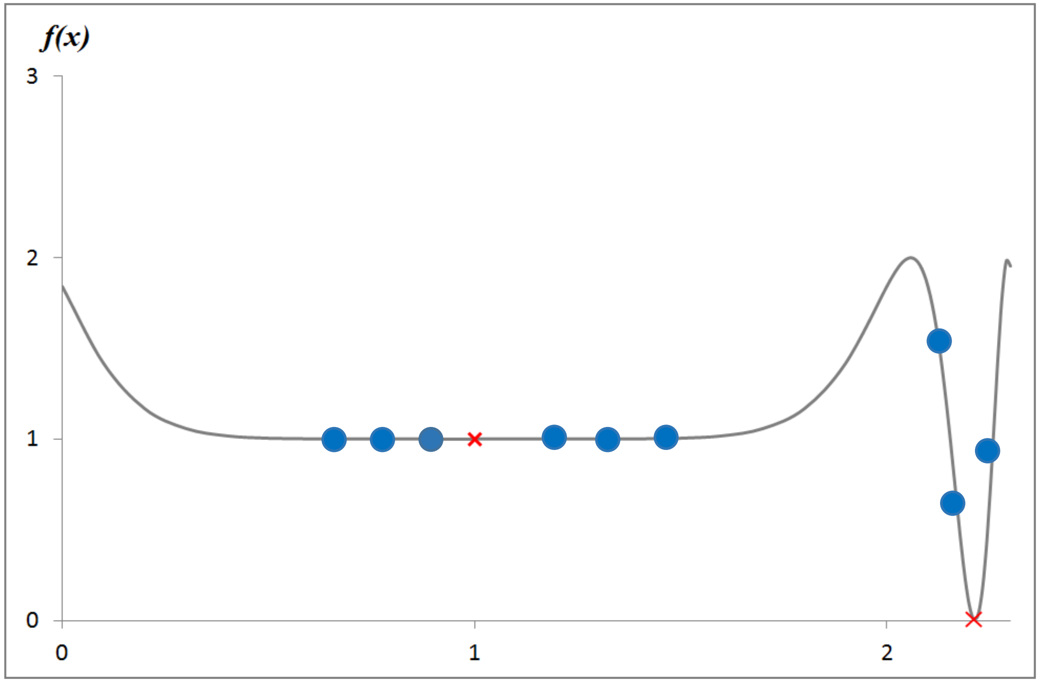
$$\min\ f(x) = \sin\left((x-1)^8\right) + 1$$

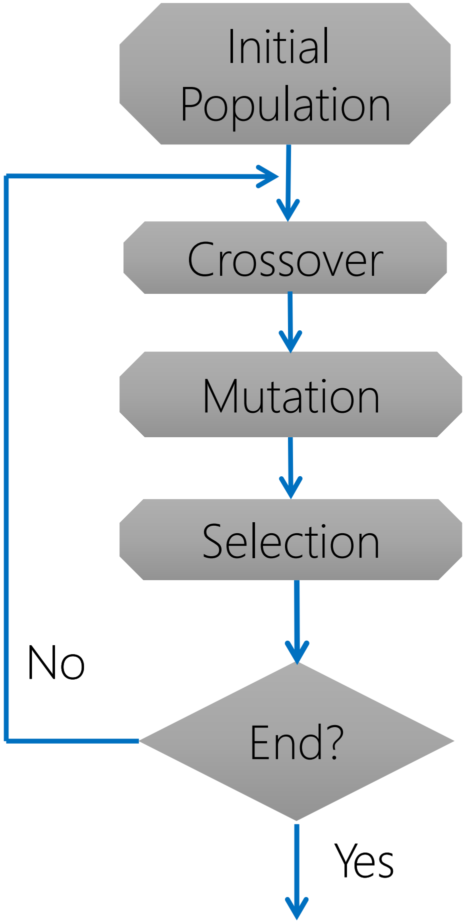




Genetic Algorithms (GAs)

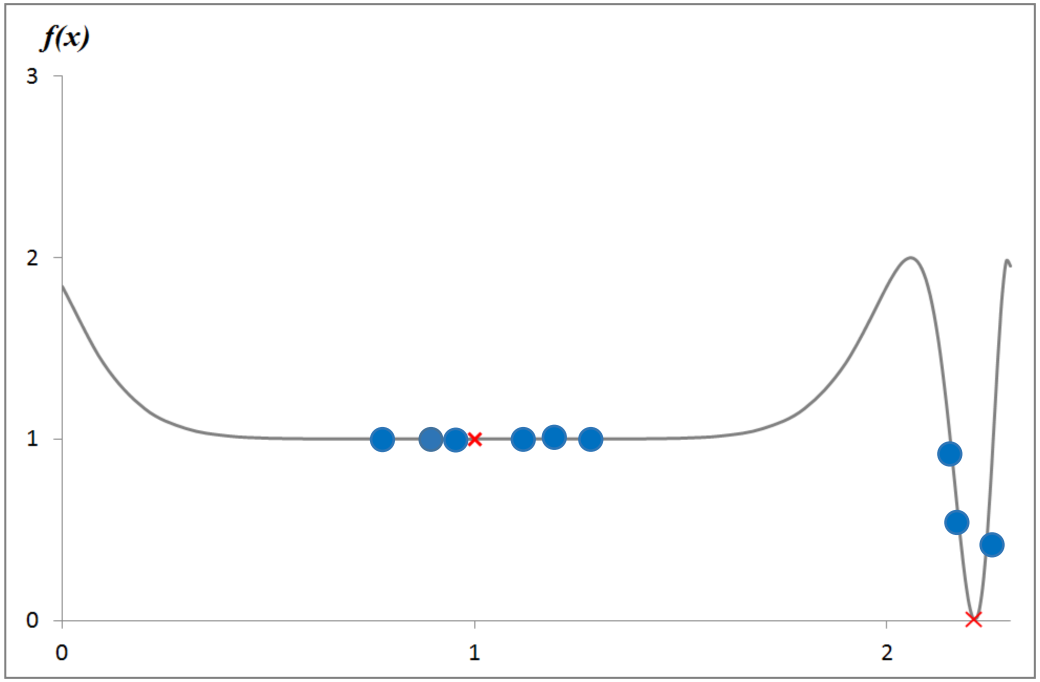
$$\min \ f(x) = \sin\left((x-1)^8\right) + 1$$

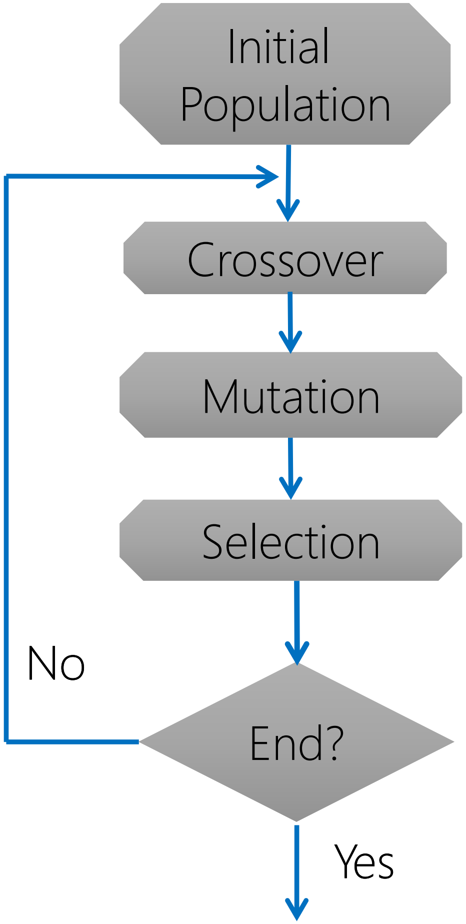




Genetic Algorithms (GAs)

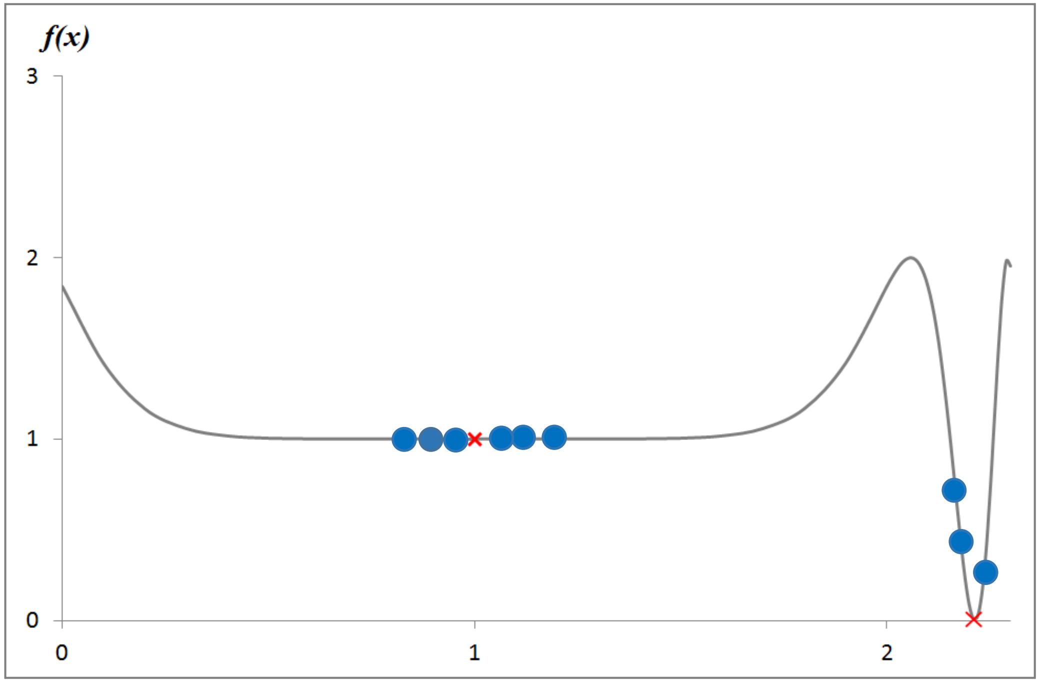
$$\min\ f(x) = \sin\left((x-1)^8\right) + 1$$

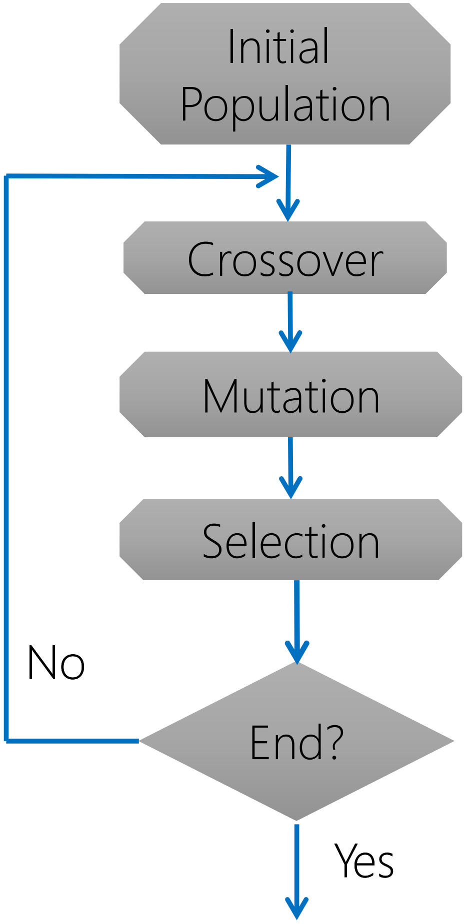




Genetic Algorithms (GAs)

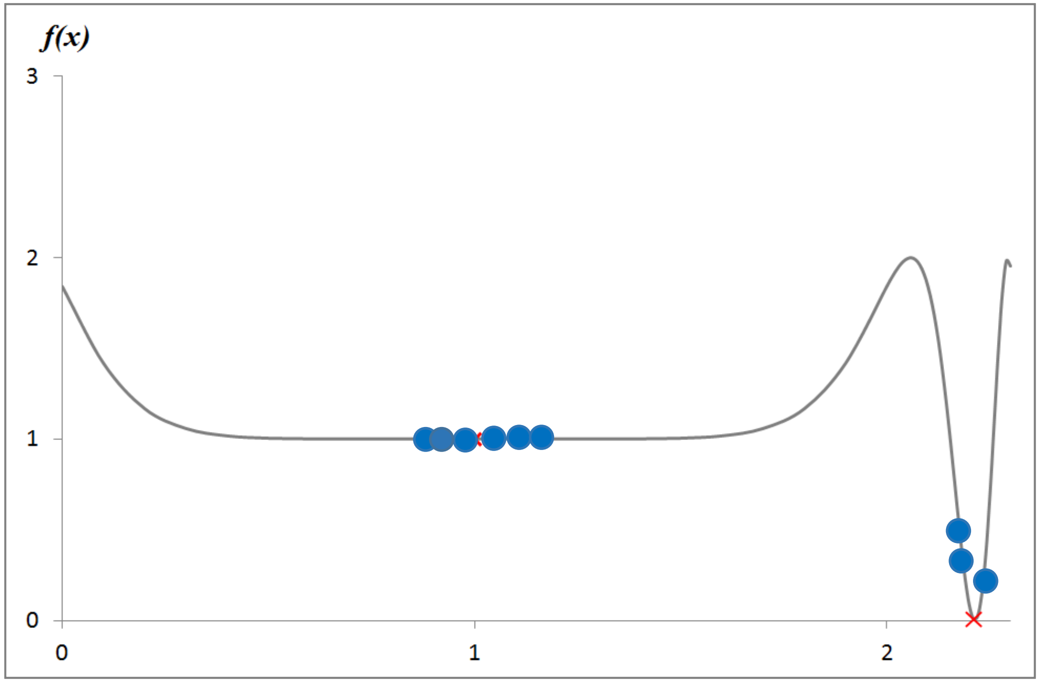
$$\min\ f(x) = \sin\left((x-1)^8\right) + 1$$

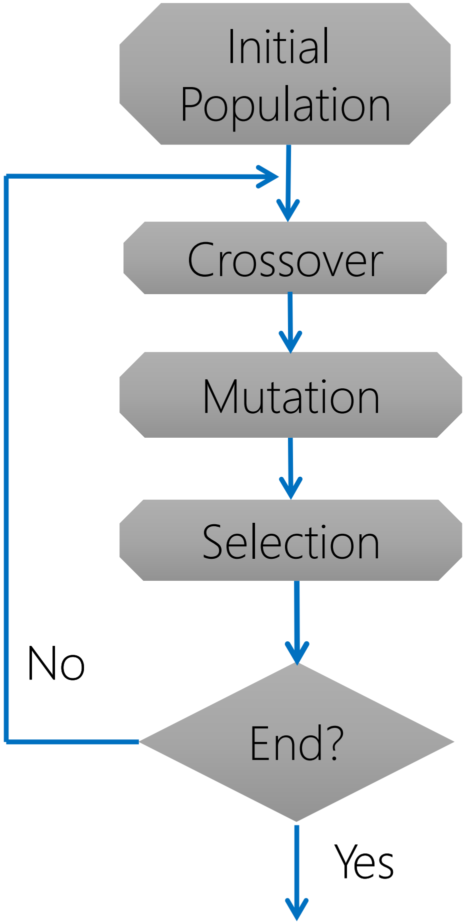




Genetic Algorithms (GAs)

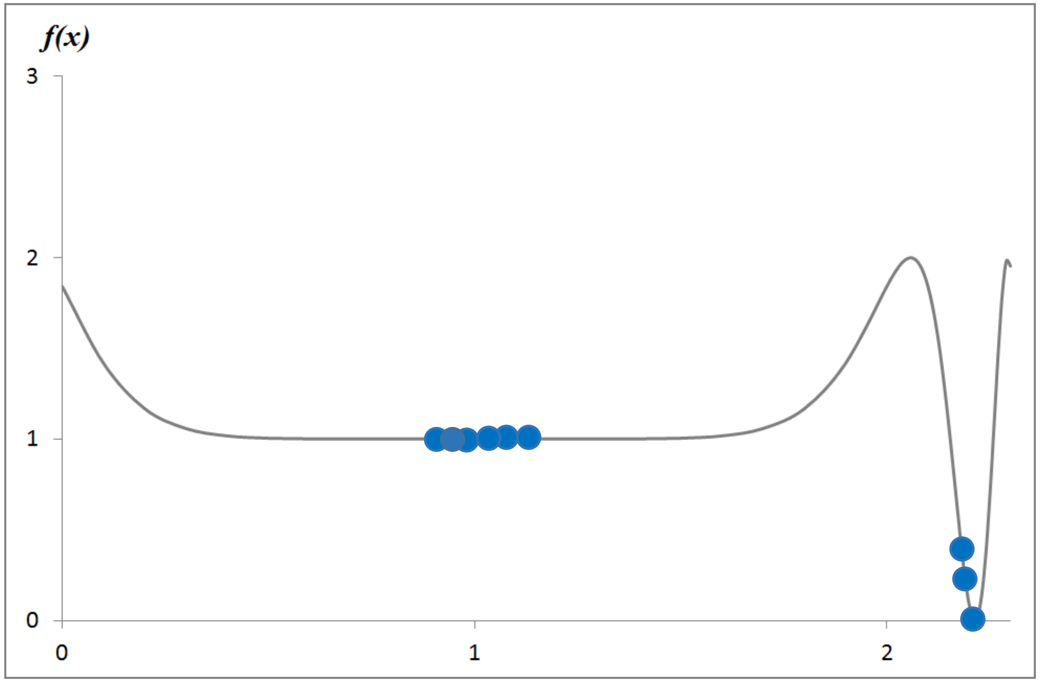
$$\min\ f(x) = \sin\left((x-1)^8\right) + 1$$

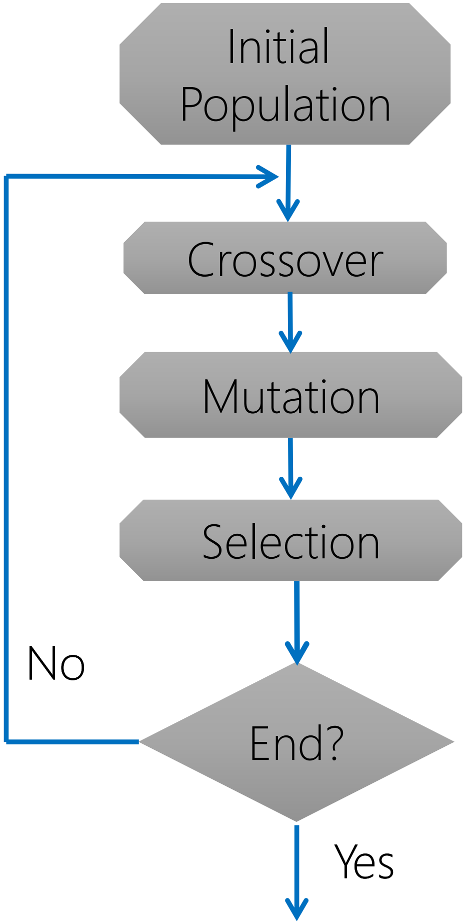




Genetic Algorithms (GAs)

$$\min\ f(x) = \sin\left((x-1)^8\right) + 1$$





Main Contributions

Search-Based Program Comprehension

Multi-Objectives Defect Prediction

Search-Based Test Data Generation

Multi-Objective Test Suite Optimization

Main Contributions

Search-Based
Program
Comprehension

Multi-Objectives
Defect Prediction

Search-Based Test
Data Generation

Multi-Objective
Test Suite
Optimization

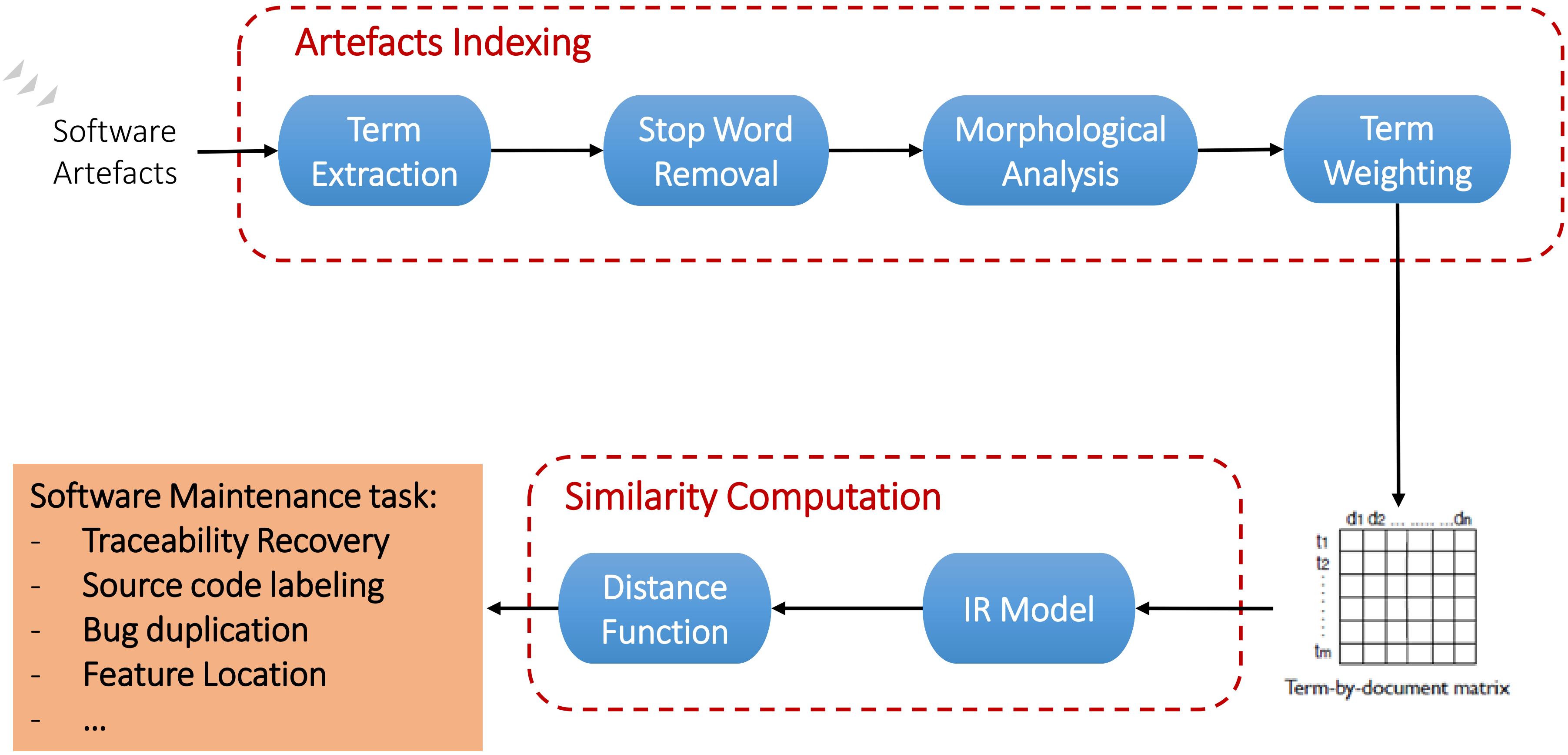
Program Comprehension

```
public class LoadConfiguration extends AbstractHandler {

IWorkbench wb = PlatformUI.getWorkbench();
IWorkbenchWindow window = wb.getActiveWorkbenchWindow();

public LoadConfiguration() {
}


@Override
public Object execute(ExecutionEvent event) throws ExecutionException {
IWorkbench wb = PlatformUI.getWorkbench();
IWorkbenchWindow window = wb.getActiveWorkbenchWindow();

IWorkbenchPage page = window.getActivePage();
IEditorPart editor = page.getActiveEditor();

//reading the instantiation variable in SCM
ResourceSet resourceSet = new ResourceSetImpl();

IFile file1;
try{
    file1 = (IFile) editor.getEditorInput().getAdapter(IFile.class);
   }catch (Exception exc){
   printError("Please select a State Chart Model", window);
   return null;
   }
SCMDiagram scd = null;
Resource scdResource = resourceSet.createResource();
try {
     scdResource.load(null);
     scd = (SCMDiagram) scdResource.getContents().get(0);
} catch (IOException e) {
     printError("Corrupted State Chart Modell file", window);
     scdResource = null;
     return null;
}
```

"Software that is not comprehended cannot be changed"- *Rajlich and Wilde* - ICPC 2002

40% to **60%** of the maintenance effort is devoted to understanding the software to be modified - *Dorfman and Thayer* – **IEEE Software Engineering 1996**

Information Retrieval



Information Retrieval

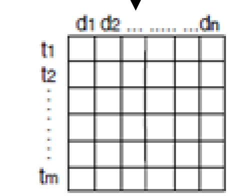
Software Artefacts

Term Extraction
- Special Chars.
- Digits
- White spaces
- …

Stop Word Removal
- Stop-word function
- Java stop-word list
- English stop-word list
- Italian stop-word list
- …

Morphological Analysis
- No Stemmer
- Porter Stemmer
- English Snowball Stemmer
- Italian Snowball Stemmer
- …

Term Weighting
- Boolean
- tf
- tf-idf
- Log(tf+1)
- Entropy
- …



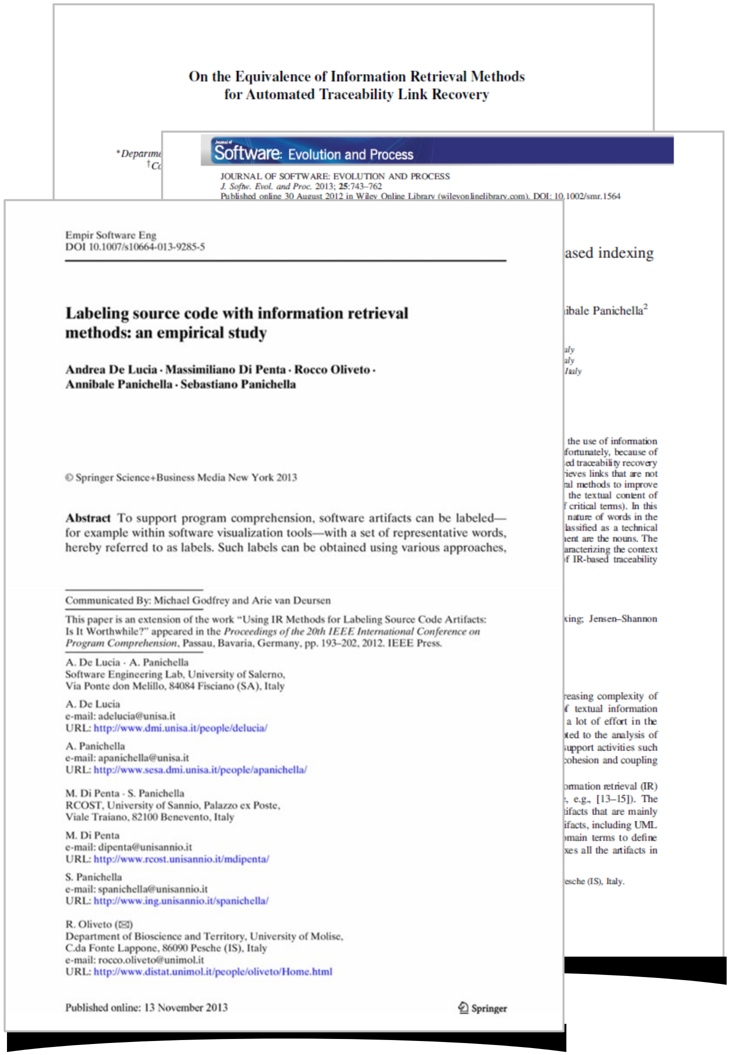
Term-by-document matrix

IR Model
- Vector Space Model
- Latent Semantic Indexing
- Latent Dirichlet Allocation
- Jensen-Shannon Divergence
- …

Distance Function
- Cosine Similarity
- Jaccard Distance
- Euclidean Distance
- Jensen-Shannon Divergence
- …

Software Maintenance task:
- Traceability Recovery
- Source code labeling
- Bug duplication
- Feature Location
- …

What is the right IR process?


On the Equivalence of Information Retrieval Methods
for Automated Traceability Link Recovery

Software: Evolution and Process

JOURNAL OF SOFTWARE: EVOLUTION AND PROCESS
J. Softw. Evol. and Proc. 2013; 25:743–762
Published online 30 August 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/smr.1564

Empir Software Eng
DOI 10.1007/s10664-013-9285-5

Labeling source code with information retrieval methods: an empirical study

Andrea De Lucia · Massimiliano Di Penta · Rocco Oliveto · Annibale Panichella · Sebastiano Panichella

© Springer Science+Business Media New York 2013

Abstract To support program comprehension, software artifacts can be labeled—for example within software visualization tools—with a set of representative words, hereby referred to as labels. Such labels can be obtained using various approaches,

Communicated By: Michael Godfrey and Arie van Deursen

This paper is an extension of the work "Using IR Methods for Labeling Source Code Artifacts: Is It Worthwhile?" appeared in the *Proceedings of the 20th IEEE International Conference on Program Comprehension*, Passau, Bavaria, Germany, pp. 193–202, 2012. IEEE Press.

A. De Lucia · A. Panichella
Software Engineering Lab, University of Salerno,
Via Ponte don Melillo, 84084 Fisciano (SA), Italy

A. De Lucia
e-mail: adelucia@unisa.it
URL: http://www.dmi.unisa.it/people/delucia/

A. Panichella
e-mail: apanichella@unisa.it
URL: http://www.sesa.dmi.unisa.it/people/apanichella/

M. Di Penta · S. Panichella
RCOST, University of Sannio, Palazzo ex Poste,
Viale Traiano, 82100 Benevento, Italy

M. Di Penta
e-mail: dipenta@unisannio.it
URL: http://www.rcost.unisannio.it/mdipenta/

S. Panichella
e-mail: spanichella@unisannio.it
URL: http://www.ing.unisannio.it/spanichella/

R. Oliveto (✉)
Department of Bioscience and Territory, University of Molise,
C.da Fonte Lappone, 86090 Pesche (IS), Italy
e-mail: rocco.oliveto@unimol.it
URL: http://www.distat.unimol.it/people/oliveto/Home.html

Published online: 13 November 2013 Springer

It is not possible to build a set of guidelines for assembling IR-based solutions for a given data set

Different dataset require different IR parameters

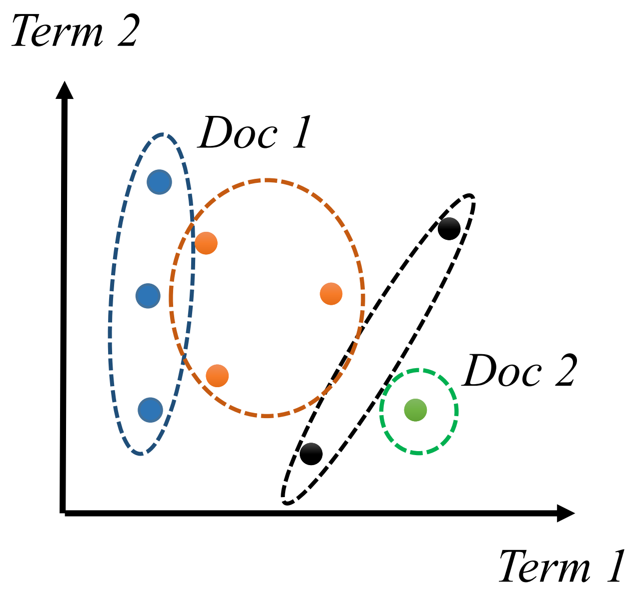
If not well calibrated, IR techniques perform worst than simple heuristics. A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella – Empirical Software Engineering

Predicting the performances?

Term Extraction	Special Chars. Digits White space
Stop Word Removal	Stop-word function Java stop-word list English stop-word list Italian stop-word list
Morphological Analysis	No Stemmer Porter Stemmer English Snowball Stemmer Italian Snowball Stemmer
Term Weighting	Boolean tf tf-idf Log(tf+1) Entropy
IR Model	LSI (k) LDA (alpha, beta, n, k)
Distance Function	Cosine Similarity Hellinger Distance

Predicting the performances?

Term Extraction	Special Chars. Digits White space
Stop Word Removal	Stop-word function Java stop-word list English stop-word list Italian stop-word list
Morphological Analysis	No Stemmer Porter Stemmer English Snowball Stemmer Italian Snowball Stemmer
Term Weighting	Boolean tf tf-idf Log(tf+1) Entropy
IR Model	LSI (k=3) LDA (alpha, beta, n, k)
Distance Function	Cosine Similarity Hellinger Distance



Predicting the performances?

Term Extraction
- Special Chars.
- Digits
- White space

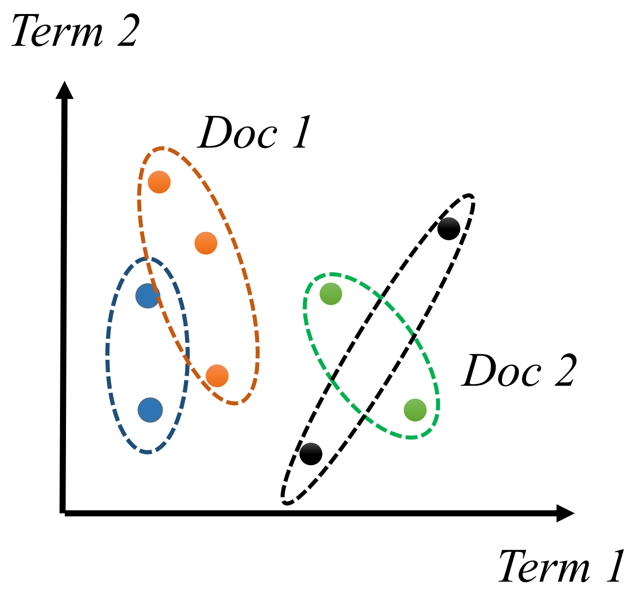
Stop Word Removal
- Stop-word function
- Java stop-word list
- English stop-word list
- Italian stop-word list

Morphological Analysis
- No Stemmer
- Porter Stemmer
- English Snowball Stemmer
- Italian Snowball Stemmer

Term Weighting
- Boolean
- tf
- tf-idf
- Log(tf+1)
- Entropy

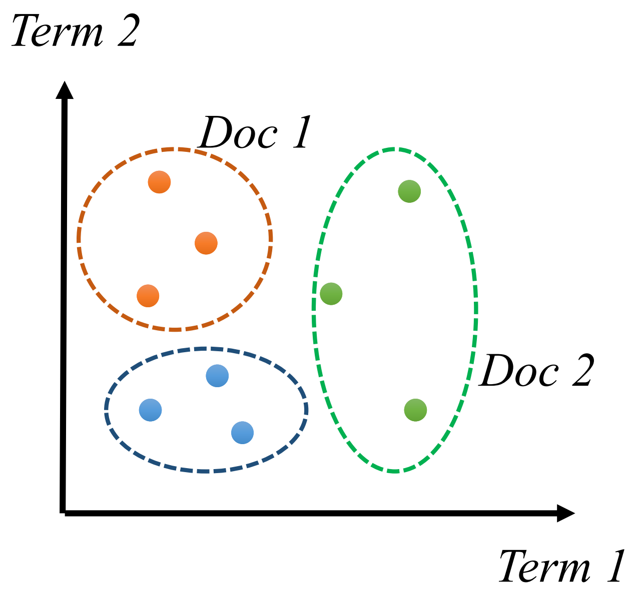
IR Model
- LSI (k=4)
- LDA (alpha, beta, n, k)

Distance Function
- Cosine Similarity
- Hellinger Distance



Predicting the performances?

Term Extraction	Special Chars. Digits White space
Stop Word Removal	Stop-word function Java stop-word list English stop-word list Italian stop-word list
Morphological Analysis	No Stemmer Porter Stemmer English Snowball Stemmer Italian Snowball Stemmer
Term Weighting	Boolean tf tf-idf Log(tf+1) Entropy
IR Model	LSI (k=4) LDA (alpha, beta, n, k)
Distance Function	Cosine Similarity Hellinger Distance



Predicting the performances?

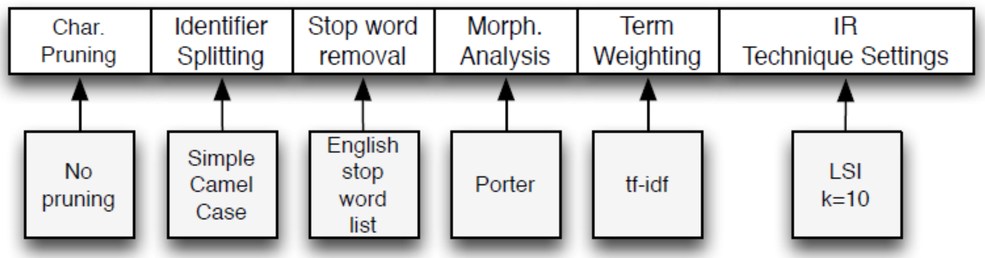
Conjecture: there is a *relationship* between quality of clusters and IR process performances

Search-Based Solution (LSI-GA)

1) **Problem Reformulation**: *Finding the IR process which maximize the quality of clusters*

2) **Solution Encoding**

X =



3) **Fitness Function**: Silhouette Coefficient

F(X)=Silhouette Coefficient (X) $= \frac{1}{n} \sum_{i=1}^{n} \frac{separation(d_i) - cohesion(d_i)}{\max\{separation(d_i), cohesion(d_i)\}}$

4) **Solver:** Genetic Algorithms

Empirical Evaluation

1) Traceability Recovery

System	Artifacts			N. Links
	Type	Number	Total	
EasyClinic	Use Case	30	77	83
	Code Classes	47		
eTour	Use Case	58	174	246
	Code Classes	116		
iTrust	Code Classes	33	149	58
	JSP	116		

2) Feature Location

System	KLOC	Files	Methods	Features
jEdit	104	503	6,413	159
JabReg	74	579	4,607	39

3) Bug Report Duplication

System	N. Bug Rep.	N. Duplication
Eclipse	225	44

Empirical Evaluation

1) <u>Traceability Recovery</u>

System	Artifacts			N. Links
	Type	Number	Total	
EasyClinic	Use Case	30	77	83
	Code Classes	47		
eTour	Use Case	58	174	246
	Code Classes	116		
iTrust	Code Classes	33	149	58
	JSP	116		

Empirical Evaluation

1) Traceability Recovery

System	Artifacts			N. Links
	Type	Number	Total	
EasyClinic	Use Case	30	77	83
	Code Classes	47		
eTour	Use Case	58	174	246
	Code Classes	116		
iTrust	Code Classes	33	149	58
	JSP	116		

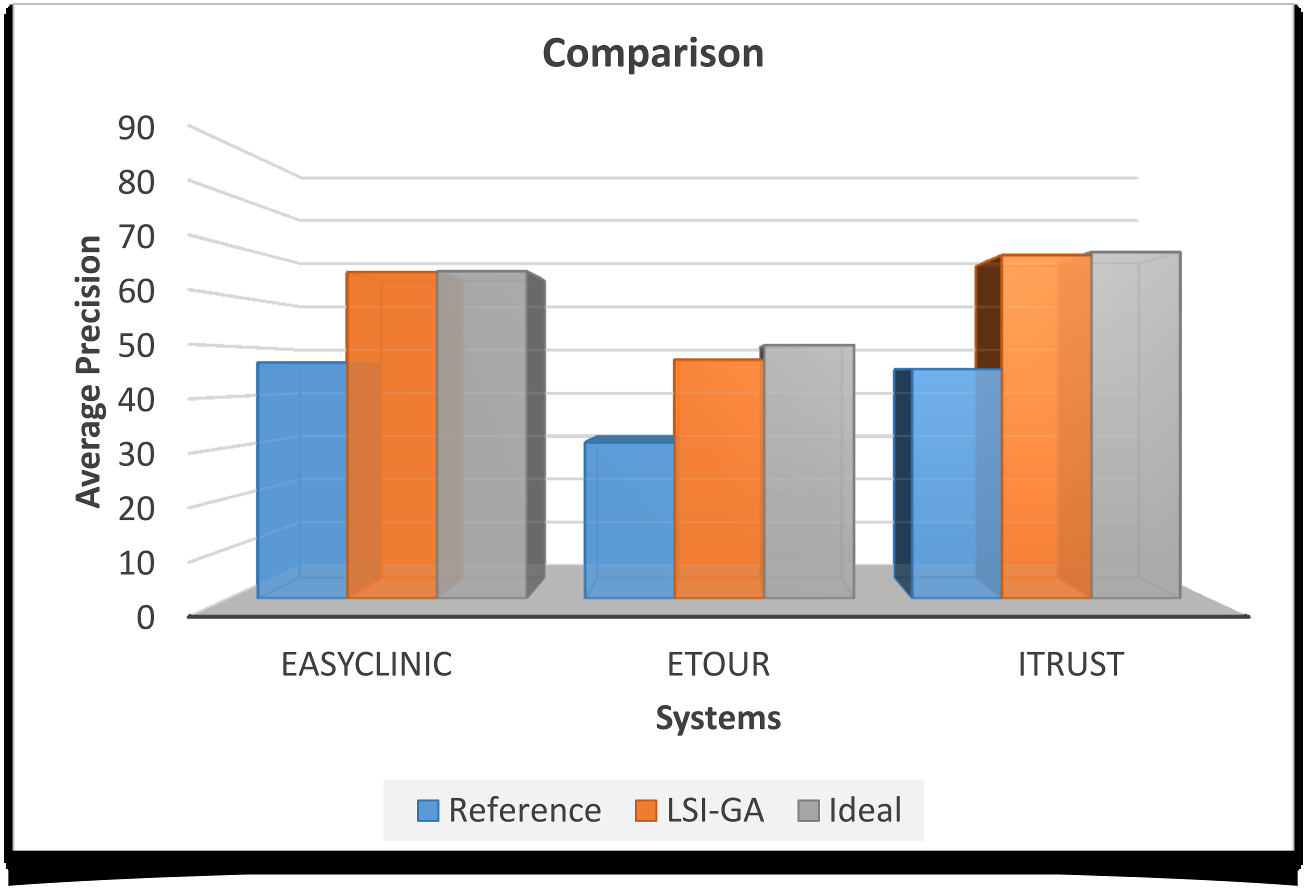
Experimented techniques:

1. LSI-GA
2. Previously published IR process
3. Ideal IR process

Performance metrics:

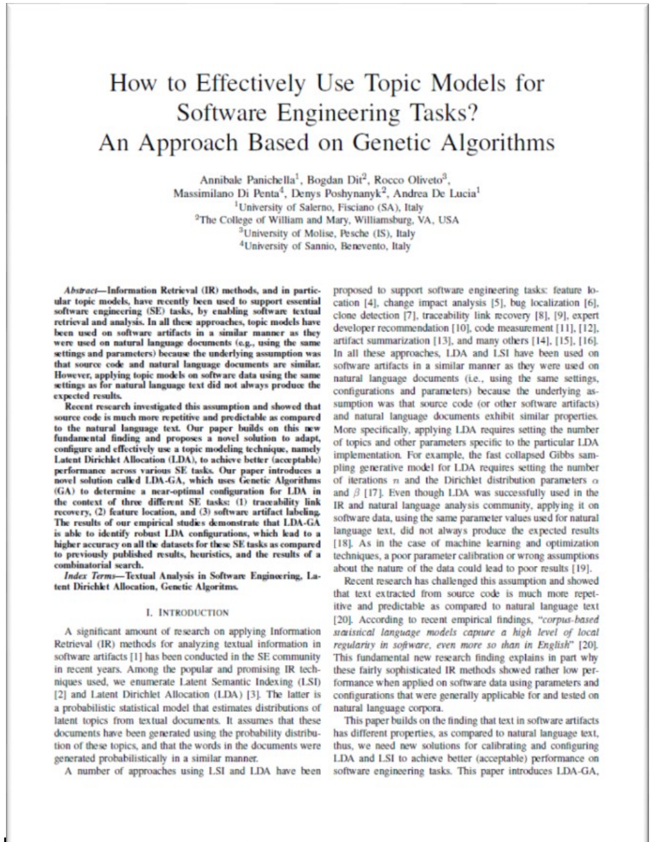
Average precision

Results



LSI-GA outperforms baseline (*p-value < 0.05*)
Ideal is statistically better than LSI-GA

Configuring LDA using GAs



How to Effectively Use Topic Models for Software Engineering Tasks? An Approach Based on Genetic Algorithms

Annibale Panichella[1], Bogdan Dit[2], Rocco Oliveto[3], Massimilano Di Penta[4], Denys Poshynanyk[2], Andrea De Lucia[1]
[1]University of Salerno, Fisciano (SA), Italy
[2]The College of William and Mary, Williamsburg, VA, USA
[3]University of Molise, Pesche (IS), Italy
[4]University of Sannio, Benevento, Italy

Abstract—**Information Retrieval (IR) methods, and in particular topic models, have recently been used to support essential software engineering (SE) tasks, by enabling software textual retrieval and analysis. In all these approaches, topic models have been used on software artifacts in a similar manner as they were used on natural language documents (e.g., using the same settings and parameters) because the underlying assumption was that source code and natural language documents are similar. However, applying topic models on software data using the same settings as for natural language text did not always produce the expected results.**

Recent research investigated this assumption and showed that source code is much more repetitive and predictable as compared to the natural language text. Our paper builds on this new fundamental finding and proposes a novel solution to adapt, configure and effectively use a topic modeling technique, namely Latent Dirichlet Allocation (LDA), to achieve better (acceptable) performance across various SE tasks. Our paper introduces a novel solution called LDA-GA, which uses Genetic Algorithms (GA) to determine a near-optimal configuration for LDA in the context of three different SE tasks: (1) traceability link recovery, (2) feature location, and (3) software artifact labeling. The results of our empirical studies demonstrate that LDA-GA is able to identify robust LDA configurations, which lead to a higher accuracy on all the datasets for these SE tasks as compared to previously published results, heuristics, and the results of a combinatorial search.

Index Terms—**Textual Analysis in Software Engineering, Latent Dirichlet Allocation, Genetic Algoritms.**

I. INTRODUCTION

A significant amount of research on applying Information Retrieval (IR) methods for analyzing textual information in software artifacts [1] has been conducted in the SE community in recent years. Among the popular and promising IR techniques used, we enumerate Latent Semantic Indexing (LSI) [2] and Latent Dirichlet Allocation (LDA) [3]. The latter is a probabilistic statistical model that estimates distributions of latent topics from textual documents. It assumes that these documents have been generated using the probability distribution of these topics, and that the words in the documents were generated probabilistically in a similar manner.

A number of approaches using LSI and LDA have been proposed to support software engineering tasks: feature location [4], change impact analysis [5], bug localization [6], clone detection [7], traceability link recovery [8], [9], expert developer recommendation [10], code measurement [11], [12], artifact summarization [13], and many others [14], [15], [16]. In all these approaches, LDA and LSI have been used on software artifacts in a similar manner as they were used on natural language documents (i.e., using the same settings, configurations and parameters) because the underlying assumption was that source code (or other software artifacts) and natural language documents exhibit similar properties. More specifically, applying LDA requires setting the number of topics and other parameters specific to the particular LDA implementation. For example, the fast collapsed Gibbs sampling generative model for LDA requires setting the number of iterations n and the Dirichlet distribution parameters α and β [17]. Even though LDA was successfully used in the IR and natural language analysis community, applying it on software data, using the same parameter values used for natural language text, did not always produce the expected results [18]. As in the case of machine learning and optimization techniques, a poor parameter calibration or wrong assumptions about the nature of the data could lead to poor results [19].

Recent research has challenged this assumption and showed that text extracted from source code is much more repetitive and predictable as compared to natural language text [20]. According to recent empirical findings, "*corpus-based statistical language models capture a high level of local regularity in software, even more so than in English*" [20]. This fundamental new research finding explains in part why these fairly sophisticated IR methods showed rather low performance when applied on software data using parameters and configurations that were generally applicable for and tested on natural language corpora.

This paper builds on the finding that text in software artifacts has different properties, as compared to natural language text, thus, we need new solutions for calibrating and configuring LDA and LSI to achieve better (acceptable) performance on software engineering tasks. This paper introduces LDA-GA,

A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, A. De Lucia
How to Effectively Use Topic Models for Software Engineering Tasks? An Approach based on Genetic Algorithms. ICSE 2013

Other works on P.C.

- A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella Labeling Source Code with Information Retrieval Methods: An Empirical Study. *Journal EMSE 2013*
- A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella ***Applying a Smoothing Filter to Improve IR-based Traceability Recovery Processes: An Empirical Investigation***. Information and Software Technology (2012).
- G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, S. Panichella ***Improving IR-based Traceability Recovery via Noun-based Indexing of Software Artifacts***. Journal of Software: Evolution and Process (2012).
- B. Dit, A. Panichella, E. Moritz, R. Oliveto, M. Di Penta, D. Poshyvanyk, A. De Lucia ***Configuring Topic Models for Software Engineering Tasks in TraceLab***. *TEFSE 2013*
- G. Bavota, A. De Lucia, R. Oliveto, A. Panichella, F. Ricci, G. Tortora ***The Role of Artefact Corpus in LSI-based Traceability Recovery***. *TEFSE 2013*
- A. Panichella, C. McMillan, E. Moritz, D. Palmieri, R. Oliveto, D. Poshyvanyk, A. De Lucia ***When and How Using Structural Information to Improve IR-Based Traceability Recovery***. *CSMR 2013*
- G. Bavota, L. Colangelo, A. De Lucia, S. Fusco, R. Oliveto and A. Panichella. ***TraceME: Traceability Management in Eclipse***. ICSM 2013

Other works on P.C.

- A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella ***Using IR Methods for Labeling Source Code Artifacts: Is It Worthwhile?*** ICPC 2012

- A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella ***Improving IR-based Traceability Recovery Using Smoothing Filters***. ICPC 2011. Best Paper Award

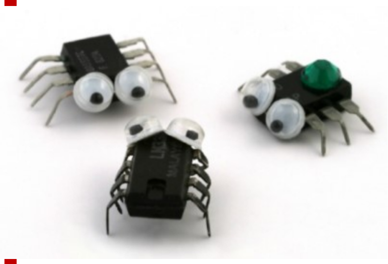
- G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, S. Panichella ***Traceability recovery using numerical analysis***. WCRE 2009.

- G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, S. Panichella ***On the Role of the Nouns in IR-based Traceability Link Recovery***. ICPC 2009.

- G. Bavota, L. Colangelo, A. De Lucia, S. Fusco, R. Oliveto and A. Panichella. ***Enhancing Traceability Management in Eclipse via Information Retrieval and User Feedback Analysis***. *ECLIPSE*

Main Contributions

Search-Based
Program
Comprehension

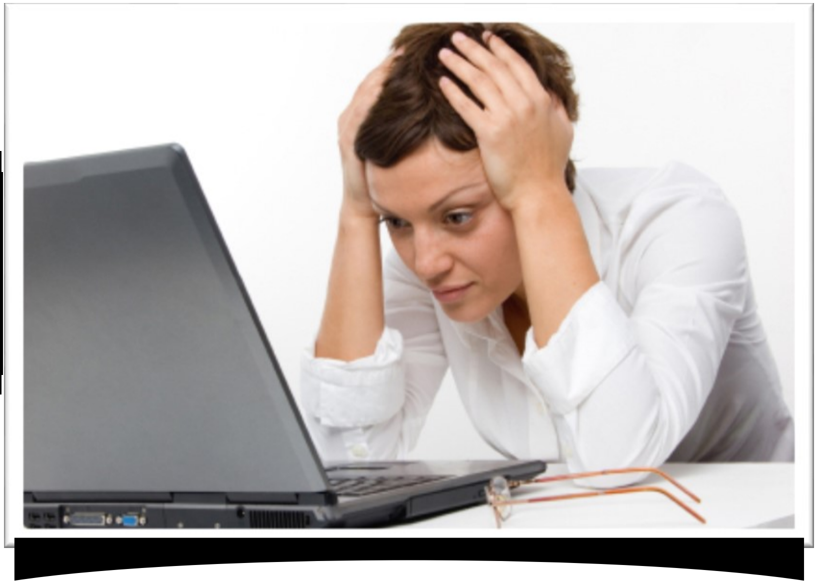


Multi-Objectives
Defect Prediction

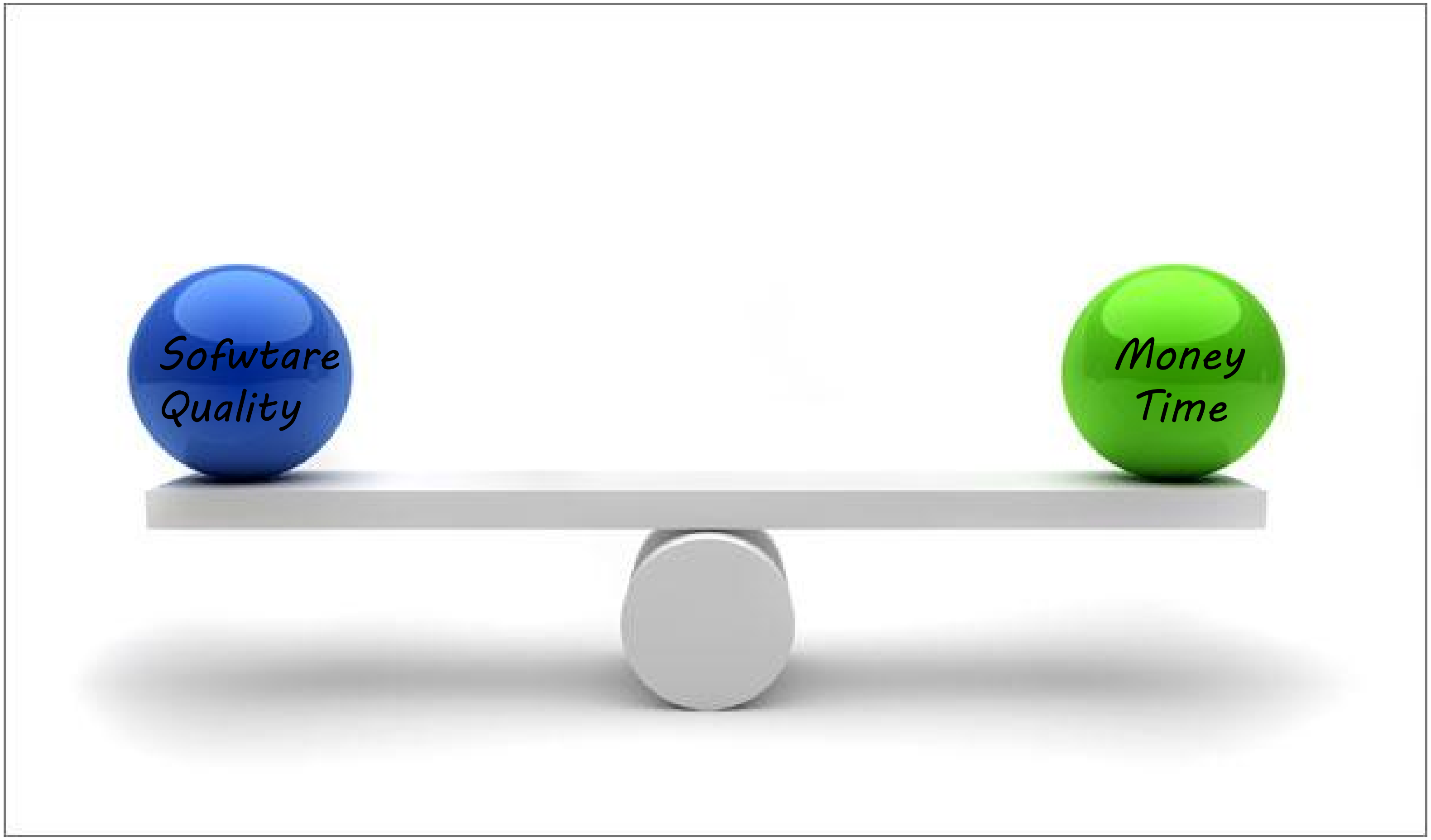
Search-Based Test
Data Generation

Multi-Objective
Test Suite
Optimization

Bugs are everywhere…



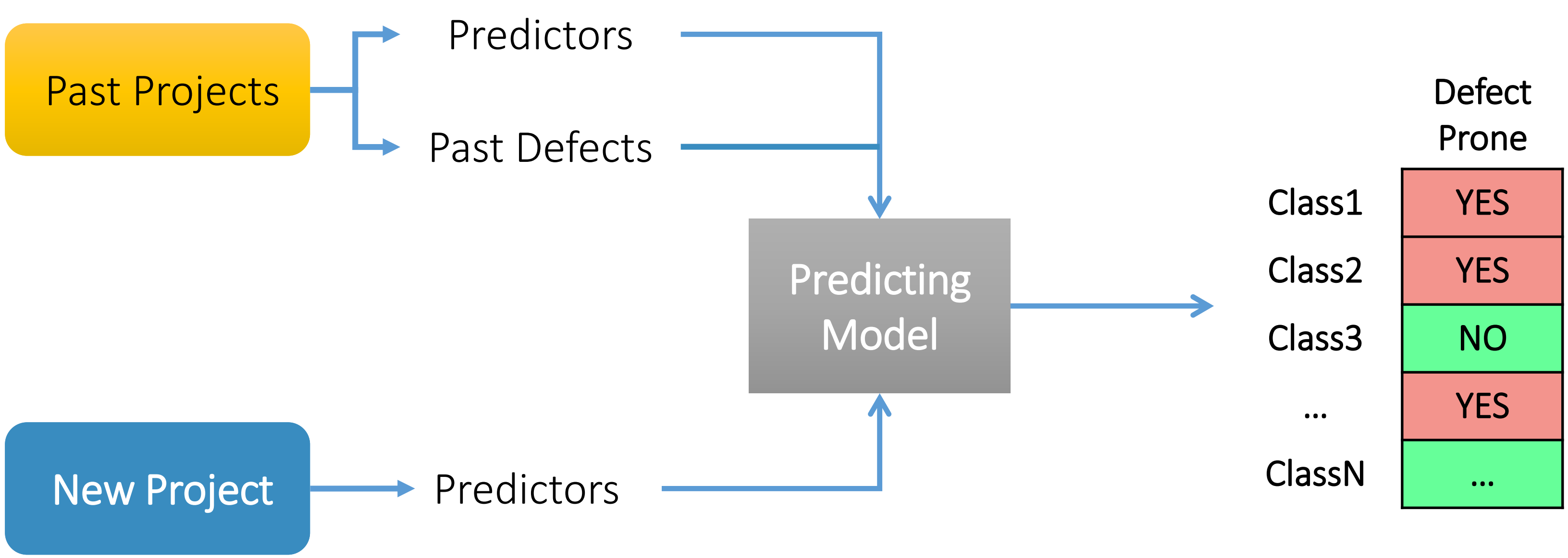
Practical Constraints



Defect Prediction

Spent more resources on components most likely to fail

Defect Prediction Methodology



Defect Prediction Methodology

All the existing predicting models work on precision and not on cost

We need COST-oriented models

Multi-objective Defect Prediction

Multi-objective Reformulation

1) **Problem Reformulation**: *Finding the logistic regression coefficients (a,b,c,…) that optimize cost and effectiveness*

$$Logit = \frac{e^{a + b\, m_{i1} + c\, m_{i2} + \dots}}{1 + e^{a + b\, m_{i1} + c\, m_{i2} + \dots}}$$

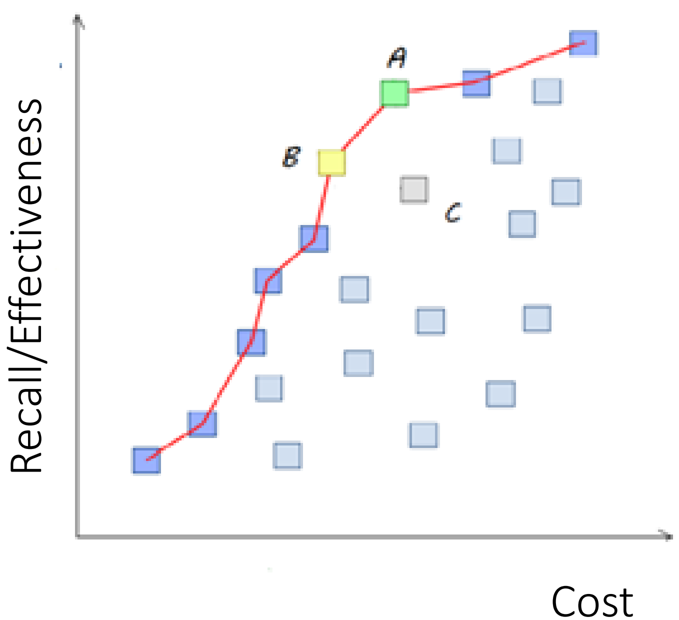
2) **Objectives Function**:

$$\begin{cases} \text{min} \quad Cost = \sum_i Pred_i \cdot LOC_i \\ \text{max} \quad \mathrm{Re}call = \sum_i Pred_i \cdot Bug_i \end{cases}$$

3) **Solver:** Multi-objective Genetic Algorithms (NSGA-II)

Multi-objective Genetic Algorithm

Multiple otpimal solutions (models) can be found



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

Multiple objectives are optimized using Pareto efficient approaches

Empirical Evaluation

Context:

Name	# Classes	#Defect-Prone Classes	% Defect-Prone Classes
Ant	745	166	22%
Camel	965	188	19%
Ivy	352	40	11%
jEdit	306	75	25%
Log4j	205	189	92%
Lucene	340	203	60%
Poi	442	281	64%
Prop	661	66	10%
Tomcat	858	77	9%
Xalan	910	898	99%

Experimented Algorithms:

1. Multi-objective cross-project Logistic Regression
2. Traditional cross-project Logistic Regression
3. Traditional within-project Logistic Regression
4. Clustering (local) cross-project defect prediction

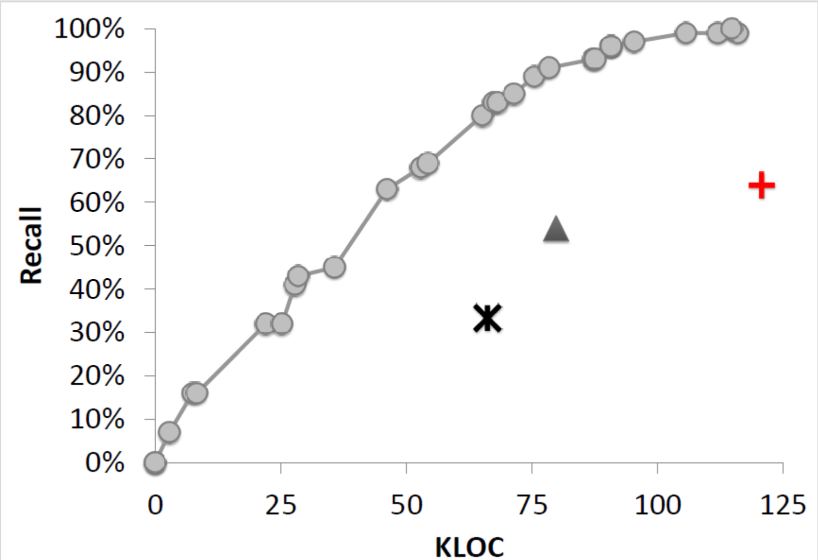
Performance metrics:

Cost = # LOC to analyze

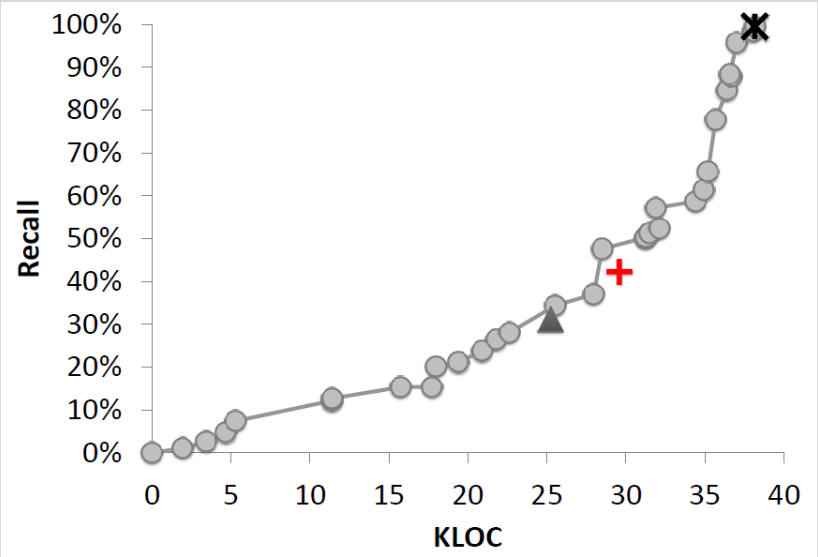
Effectiveness/Recall = % defect-prone classes identified

Results

jEdit



Log4j



Multi-Objective Logistic + Single-Objective Logistic ▲Clustering Logistic ✖Within Project Logistic

Multi-Objective Defect Prediction

Multi-Objective Cross-Project Defect Prediction

Gerardo Canfora[1], Andrea De Lucia[2], Massimiliano Di Penta[1],

SOFTWARE TESTING, VERIFICATION AND RELIABILITY
Softw. Test. Verif. Reliab. 0000; **00**:1–26
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/stvr

Defect Prediction as a Multi-Objective Optimization Problem

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

[1] *University of Sannio, Via Traiano, 82100 Benevento, Italy*
[2] *University of Salerno, Via Ponte don Melillo, 84084 Fisciano (SA), Italy*
[3] *University of Molise, Contrada Fonte Lappone, 86090 Pesche (IS), Italy*

SUMMARY

Approaches for defect prediction aim at identifying software entities having a high likelihood to exhibit fault, and that should therefore be better analyzed and tested before release. Existing defect prediction approaches only allow to pursue an implicit compromise between cost (e.g., inspection cost) and effectiveness (e.g., ability to identify defect prone software entities). However, cost and effectiveness are very conflicting objectives, and recent work showed that, when considering cost and effectiveness as separate objectives, prediction models turn out to be suitable for cross-project defect prediction, despite previous studies indicated this was challenging due to data heterogeneity.
In this paper we formalize the defect prediction problem as a multi-objective optimization problem. Specifically, we propose an approach, coined as MODEP (Multi-Objective DEfect Predictor), for defect prediction based on multi-objective forms of machine learning techniques—logistic regression and decision trees specifically—trained using a genetic algorithm. The multi-objective approach allows software engineers to choose predictors achieving a specific compromise between the number of likely defect-prone classes, or the number of faults that the analysis would likely discover (effectiveness), and LOC to be analyzed/tested (which can be considered as a proxy of the cost of code inspection).
Results of an empirical evaluation on 10 datasets from the PROMISE repository indicate the quantitative superiority of the multi-objective approaches with respect to single-objective predictors, and its capability to suggest software engineers the most suitable rules given the desired cost-effectiveness target. Also, the proposed approach outperforms an alternative approach for cross-project prediction, based on local prediction upon clusters of similar classes.

Received …

KEY WORDS: Defect prediction; multi-objective optimization; cost-effectiveness; cross-project defect prediction.

1. INTRODUCTION

Defect prediction models aim at identifying likely defect-prone software components, in order to prioritize Quality Assurance (QA) activities. The main reason why such models are required can be found in the limited time or resources available, reason for which QA teams have to focus their attention on a subset of software entities only, trying to maximize the number of discovered defects. Existing defect prediction models try to identify defect-prone artifacts based on product or process metrics. For example, Basili *et al.* [1] and Gyimothy *et al.* [2] use Chidamber and Kemerer (CK) metrics [3] to build defect prediction models based on logistic regression or neural networks. Moser *et al.* [4] use process metrics, e.g., related to the number and kinds of changes occurred on software artifacts. Ostrand *et al.* [5] and Kim *et al.* [6] perform prediction based on knowledge about previously occurred faults. Also, Kim *et al.* [7] used their SZZ algorithm [8, 9] to identify

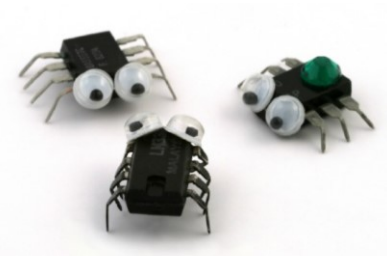
Prepared using stvrauth.cls *[Version: 2010/05/13 v2.00]*

G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella
Multi-Objective Cross-Project Defect Prediction. ICST 21013

G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella
Defect Prediction as Multi-Objective Optimization Problem.
Submitted as Special Issue on Journal STVR

Main Contributions

Search-Based Program Comprehension

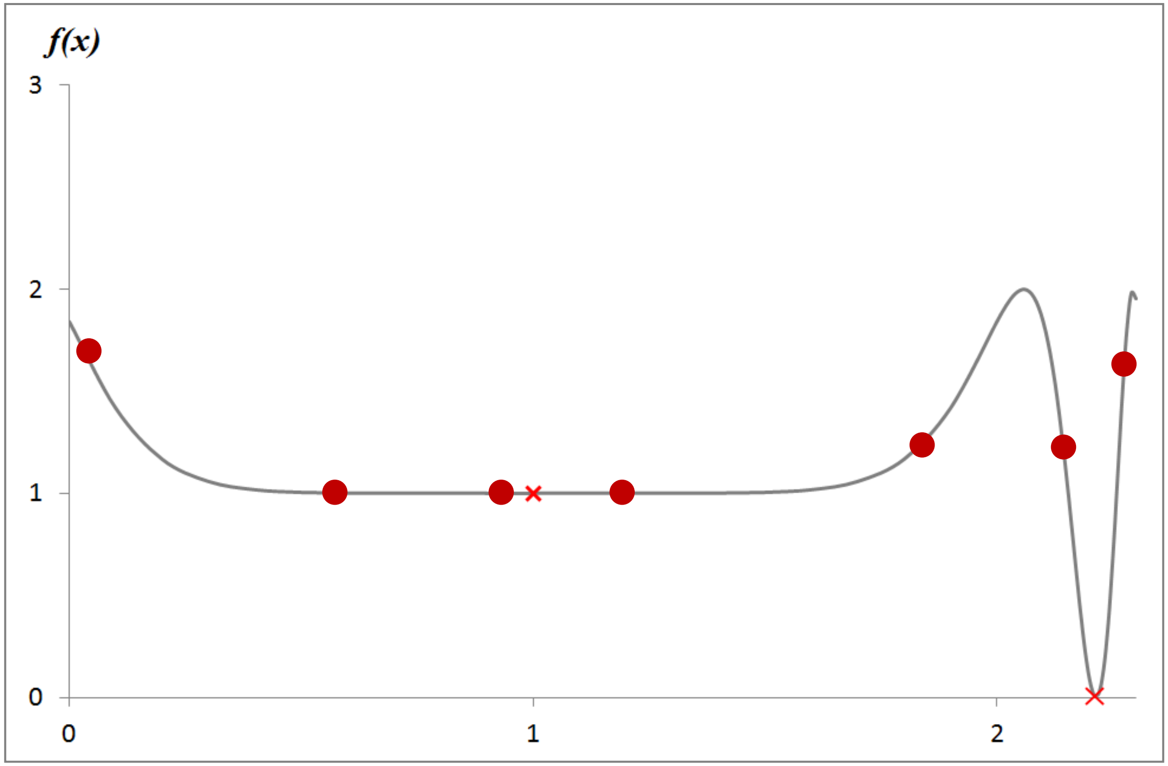


Multi-Objectives Defect Prediction

Search-Based Test Data Generation

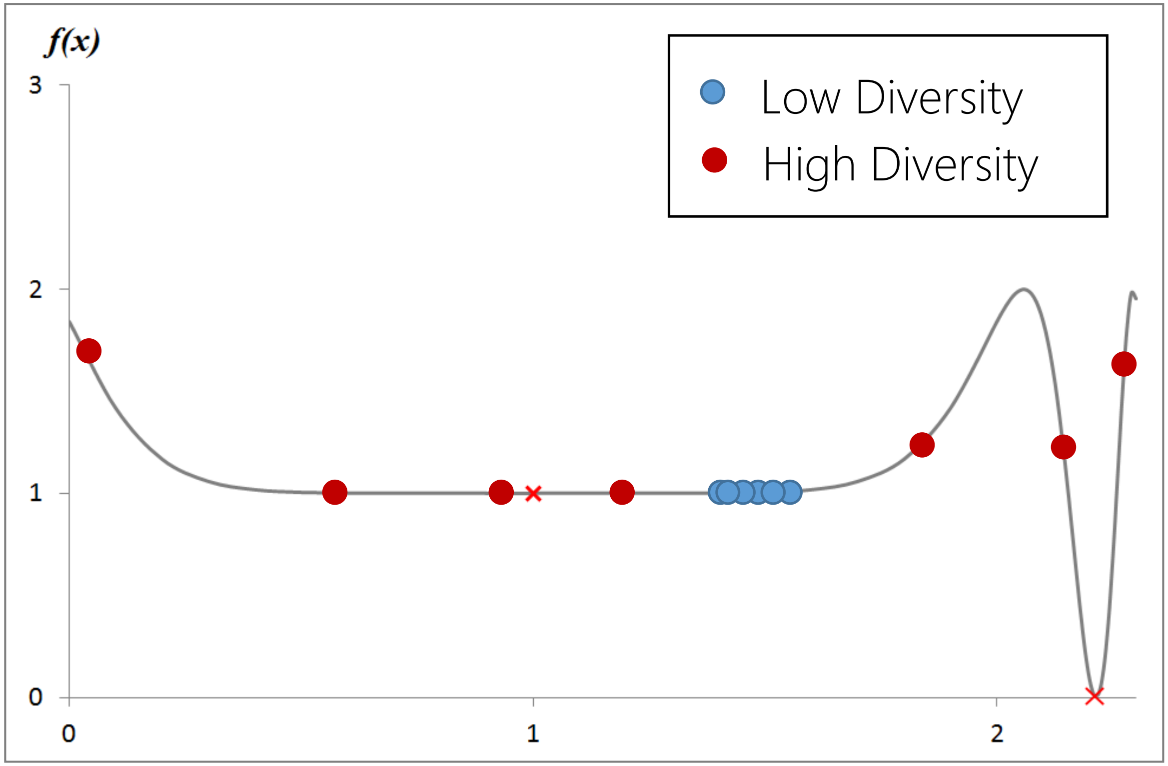
Multi-Objective Test Suite Optimization

GAs in Software Testing



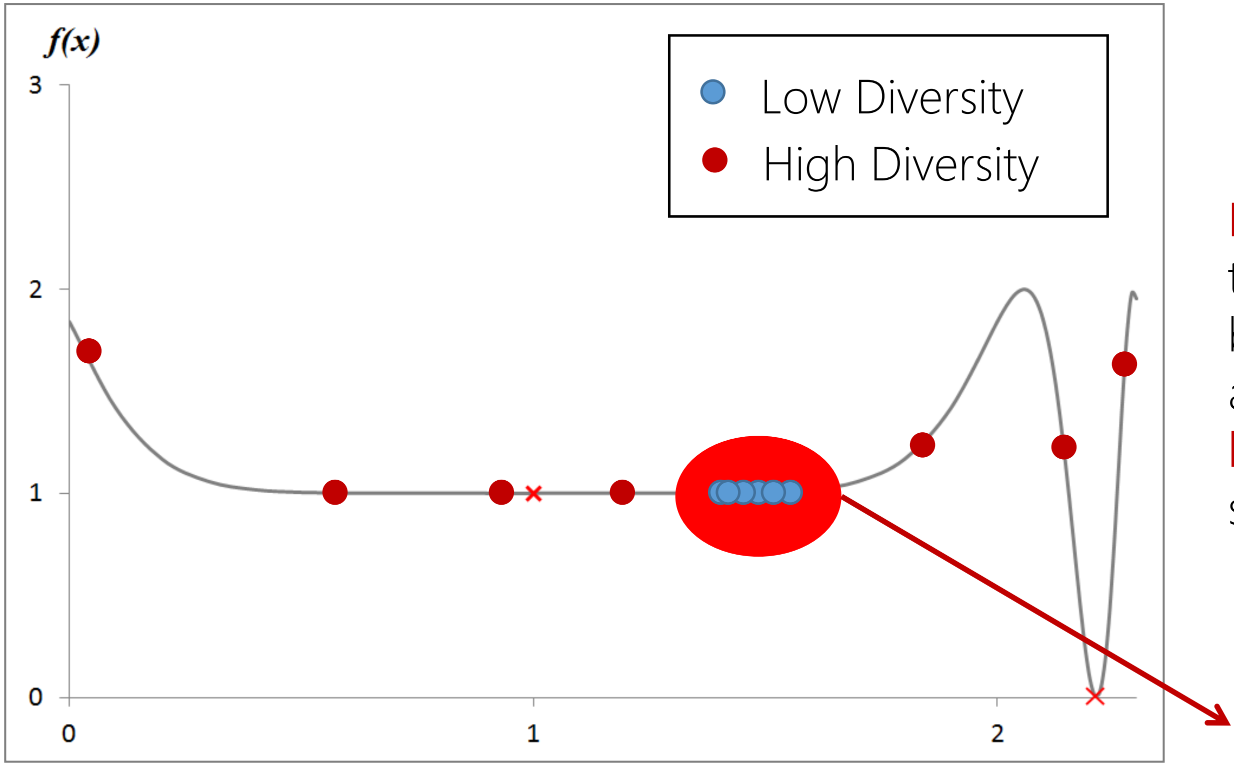
Diversity is essential to the genetic algorithm because it enables the algorithm to search a larger region of the space.

GAs in Software Testing



Diversity is essential to the genetic algorithm because it enables the algorithm to search a larger region of the space.

GAs in Software Testing



Population drift

Diversity is essential to the genetic algorithm because it enables the algorithm to search a larger region of the space.

Triangle Program

```
public class Triangle {

    public String check (double a, double b,
                                          double c){
1.      if(a == b)
            {
2.          if(a == c)
3.                      return 'equilater';
            else
4.                      return 'isoscele';
            }
        else
            {
5.          if(a == c || b == c)
6.                      return 'isoscele';
            else
7.                      return 'scalene';
            }
     }
}
```

Search-based approach

```
public class Triangle {

     public String check (double a, double b,
                                          double c){
1.    if(a == b)
          {
2.         if(a == c)
3.                   return 'equilater';
           else
4.                   return 'isoscele';
          }
      else
          {
5.         if(a == b || a == c || b == c)
6.                   return 'isoscele';
           else
7.                   return 'scalene';
          }
     }
}
```

Search-based approach

```
public class Triangle {

    public String check (double a, double b,
                                        double c){
1.      if(a == b)
           {
2.         if(a == c)
3.                 return 'equilater';
           else
4.                 return 'isoscele';
           }
        else
           {
5.         if(a == b || a == c || b == c)
6.                 return 'isoscele';
           else
7.                 return 'scalene';
           }
    }
}
```

Branch distance

(a == b) -> abs(a - b)

(a == c) -> abs(a - c)

min f(a,b,c) = 2 * abs(a - b) + + abs(a - c)

Search-based approach

```
public class Triangle {

    public String check (double a, double b,
                                  double c){
1.      if(a == b)
          {
2.         if(a == c)
3.                 return 'equilater';
           else
4.                 return 'isoscele';
          }
        else
          {
5.         if(a == b || a == c || b == c)
6.                 return 'isoscele';
           else
7.                 return 'scalene';
          }
    }
}
```

Branch distance

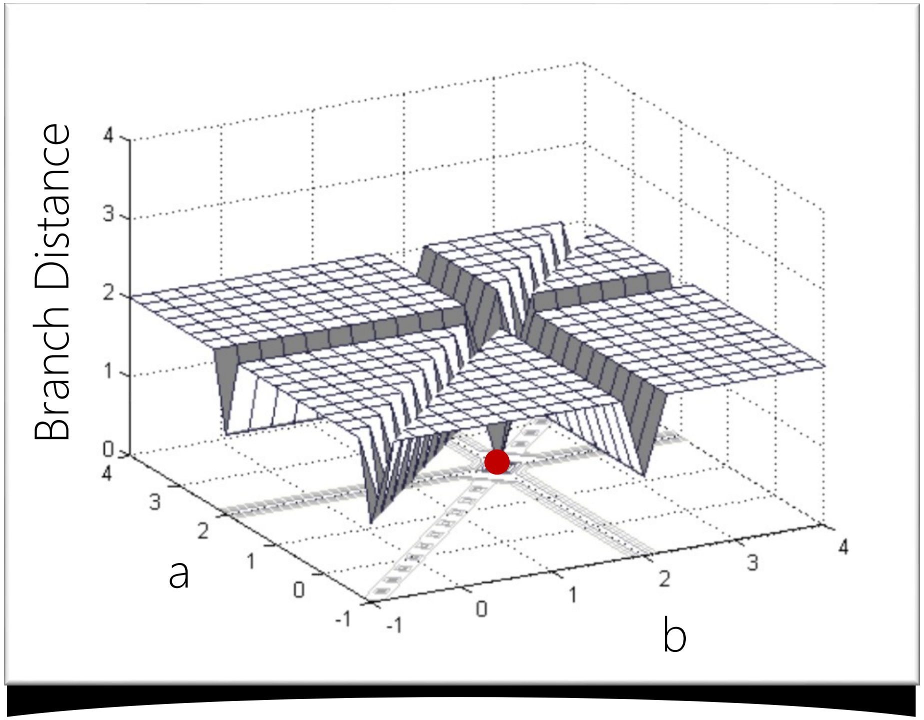
```
(a == b) -> abs(a - b)
(a == c) -> abs(a - c)
```

min f(a,b,c) = 2 * abs(a - b) + + abs(a - c)

```
Test Case 4
Triangle t= new Triangle();
String s=t.check(2,2,2)
```

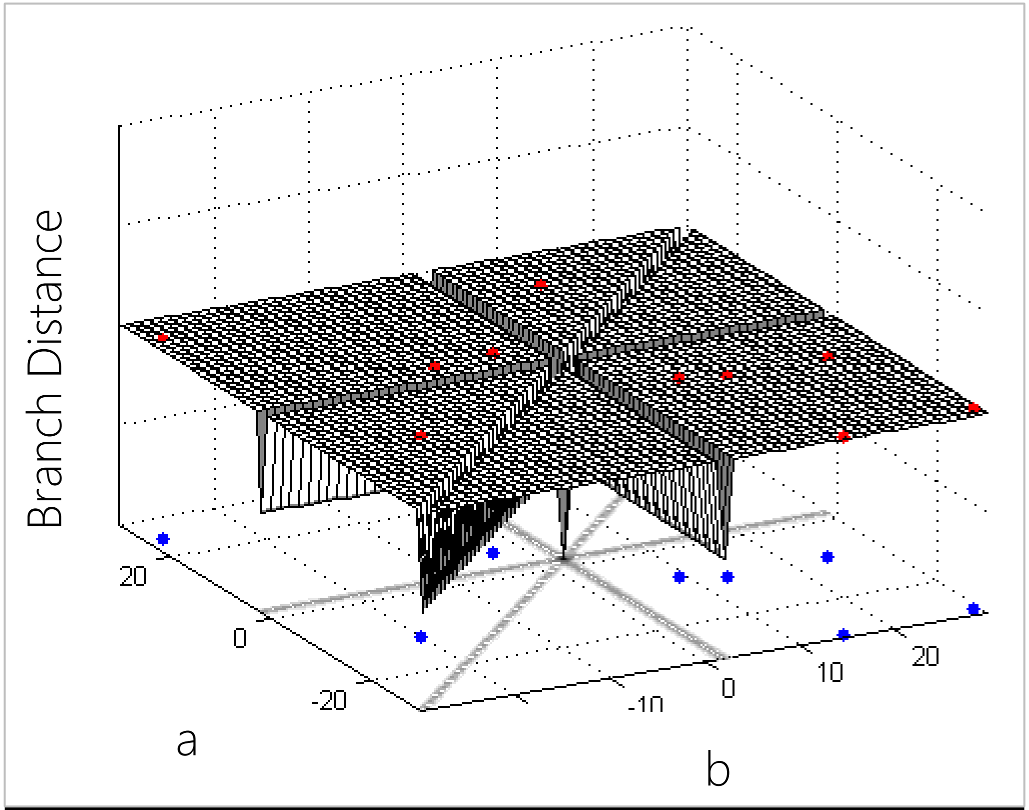
Triangle Program

c=2   a, b ∈ [-1;4]



1) Flat seach space
2) Several Local optimal
3) Only one global optimum

GAs Simulation

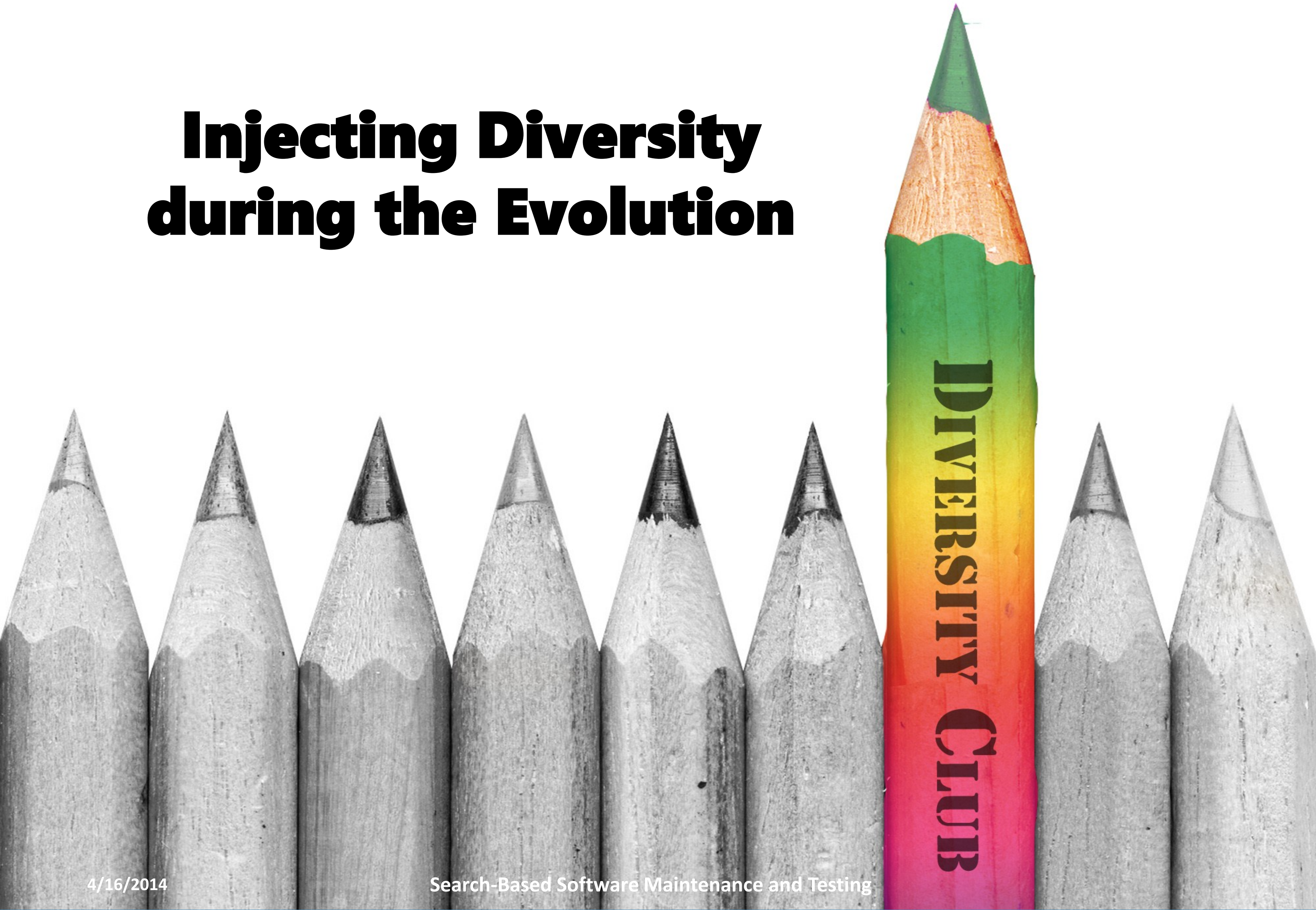


a, b ϵ [-30;30]
c=2

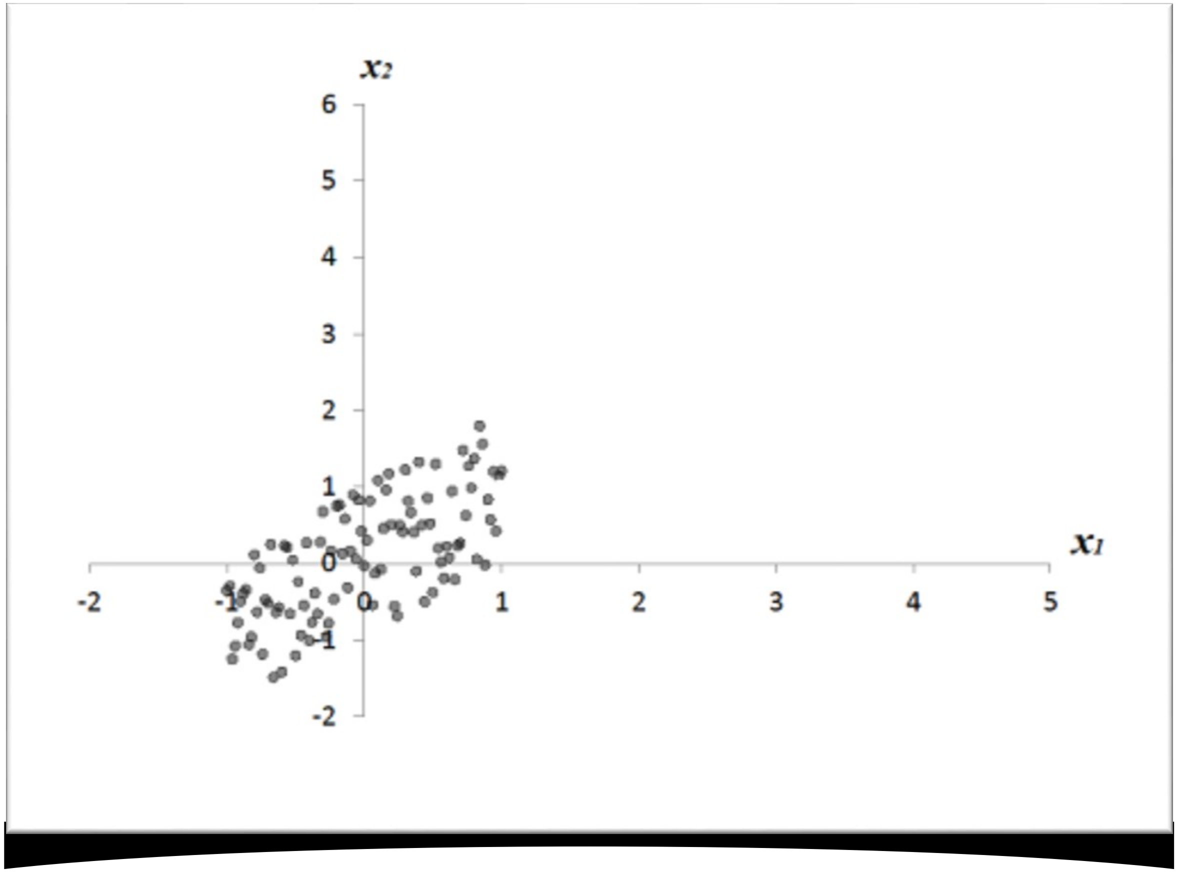
Mutation Rate = 0.10
Population = 50
Crossover = single-point

Premature convergence (genetic drift)

Injecting Diversity during the Evolution

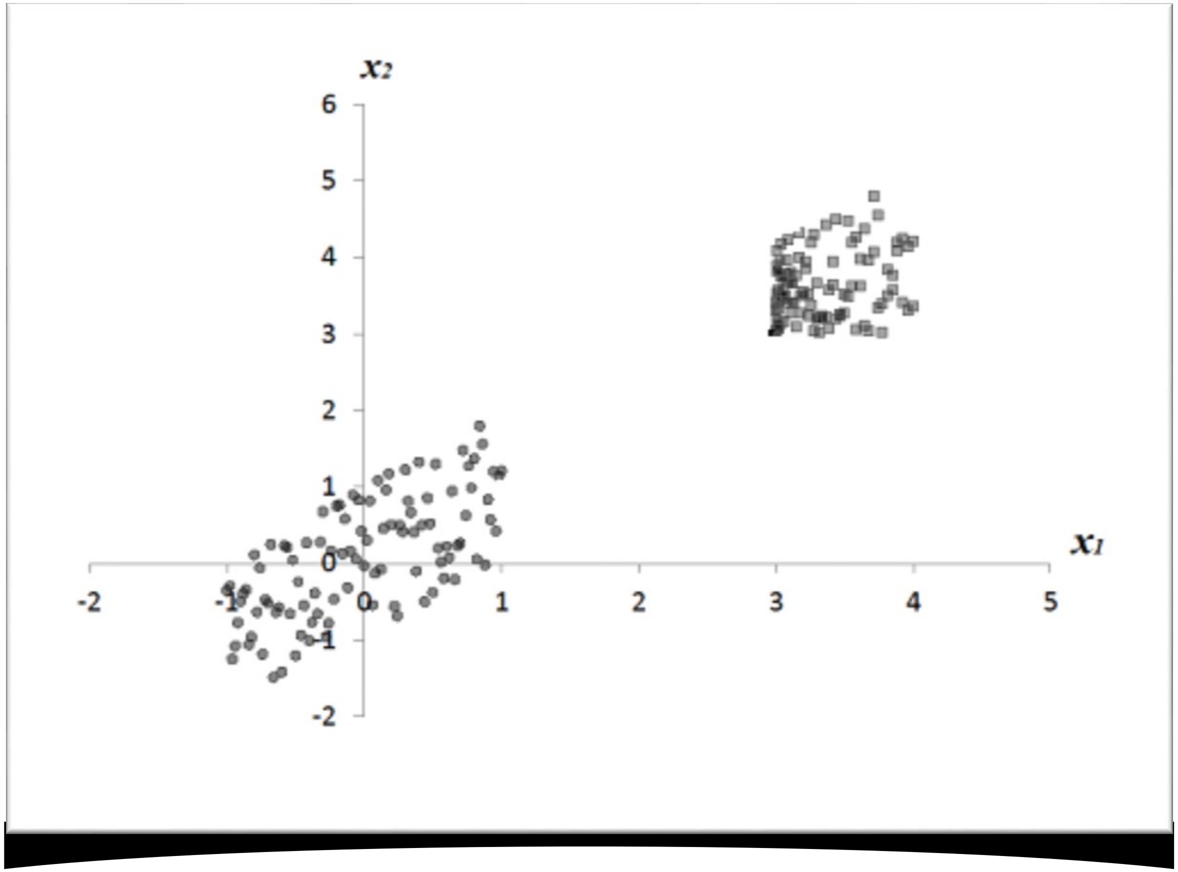


What is the evolution direction?



P(t) = Population at generation t

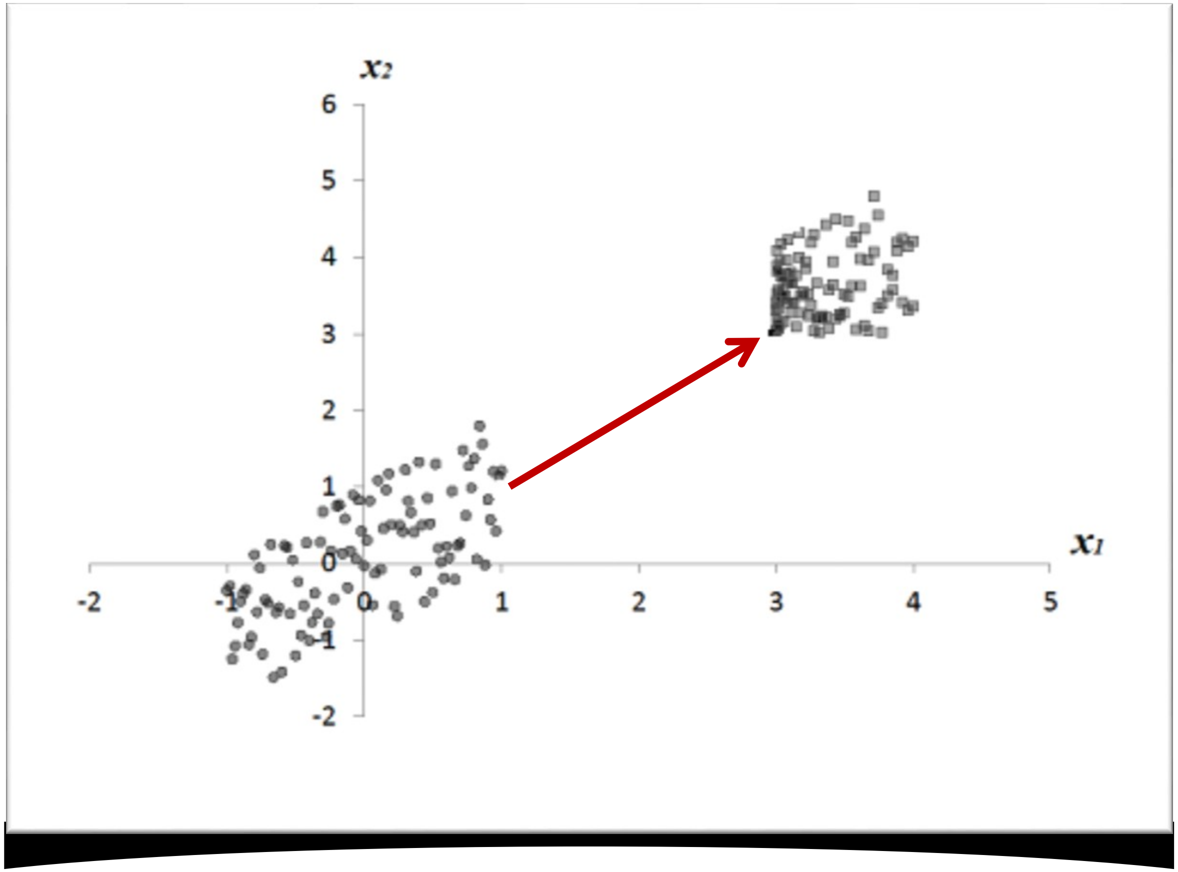
What is the evolution direction?



P(t) = Population at generation t

P(t+k) = Population after k generations

What is the evolution direction?

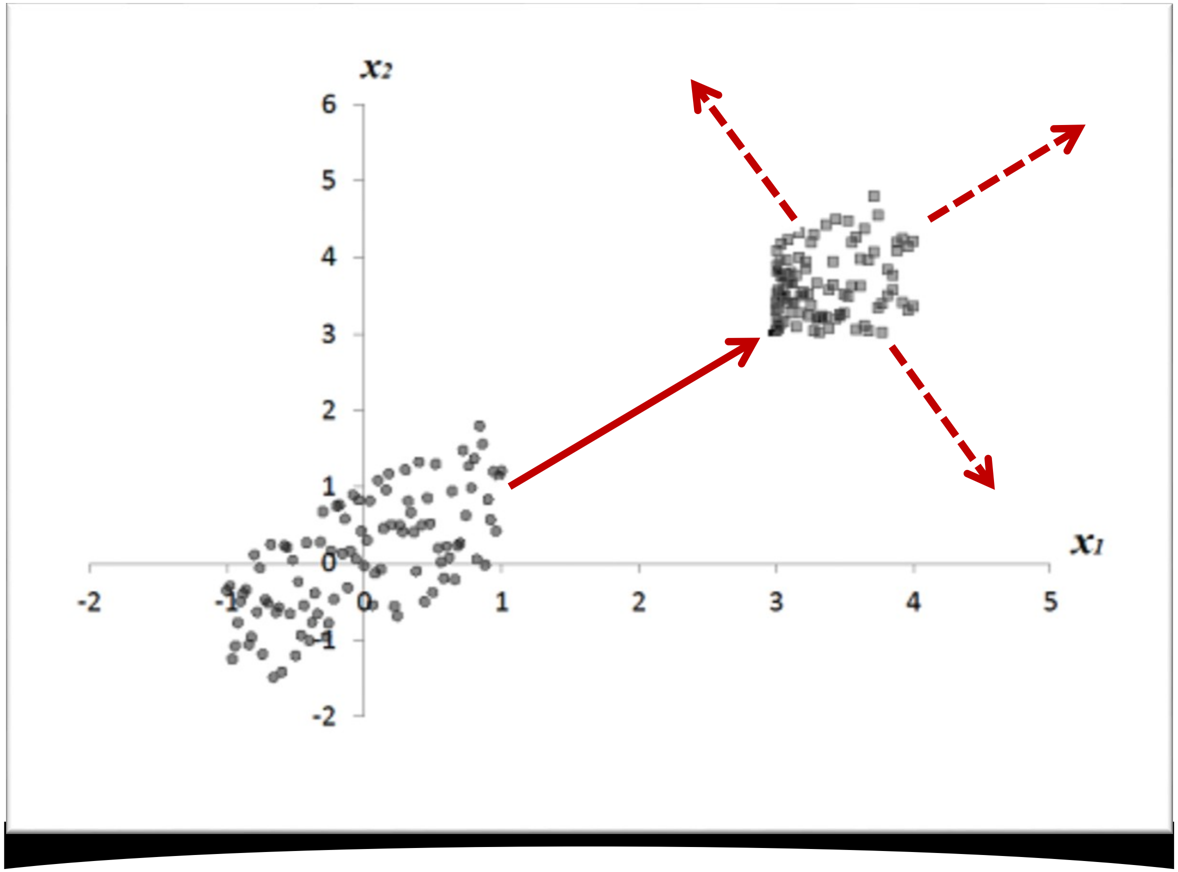


P(t) = Population at generation t

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

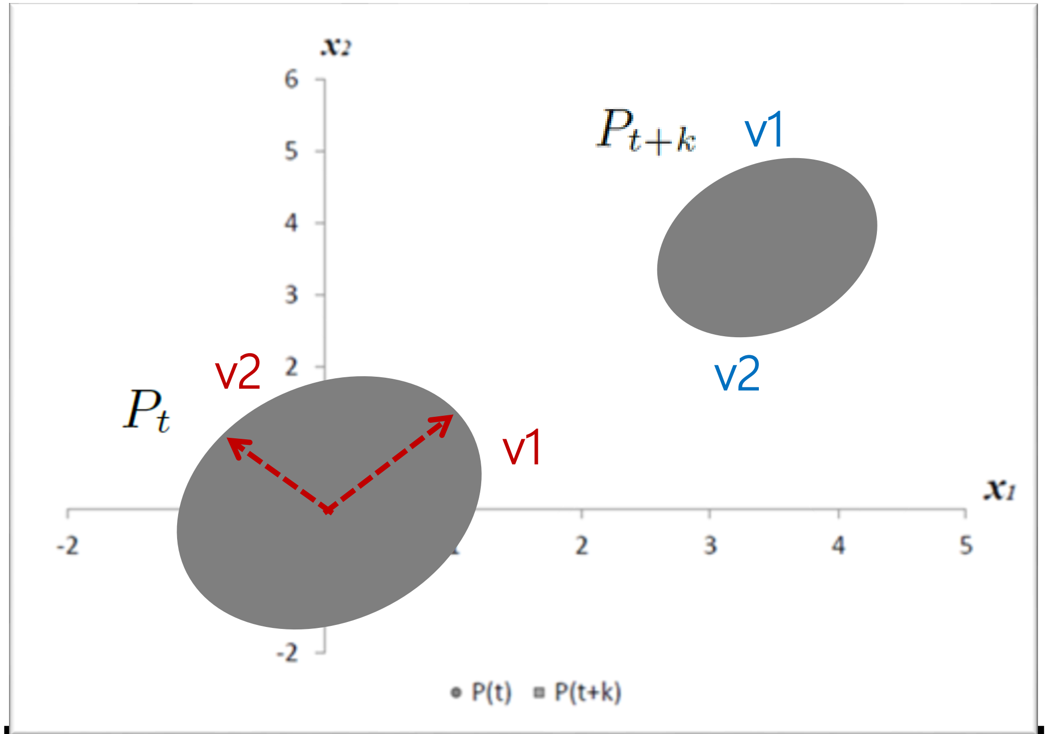
How? Singual Value Decomposition

Population at generation t

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

Population at generation t + k

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

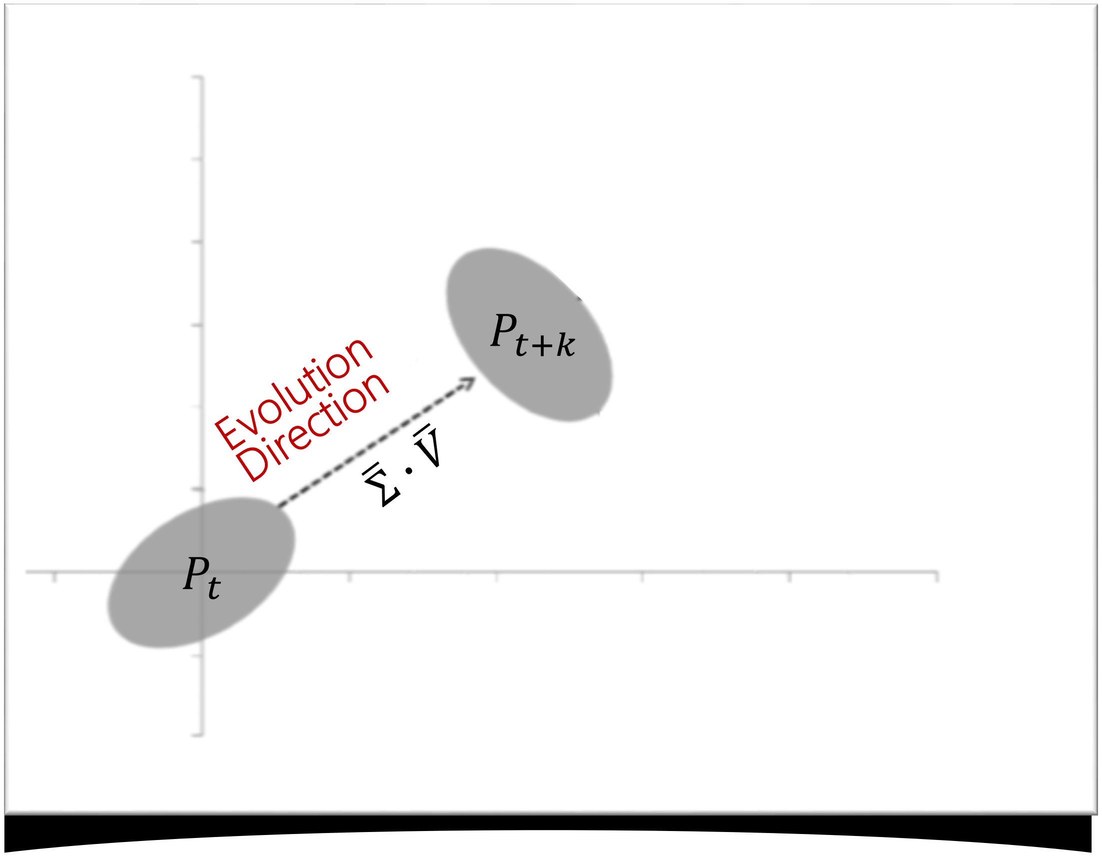


The currect evolution direction is proportional to

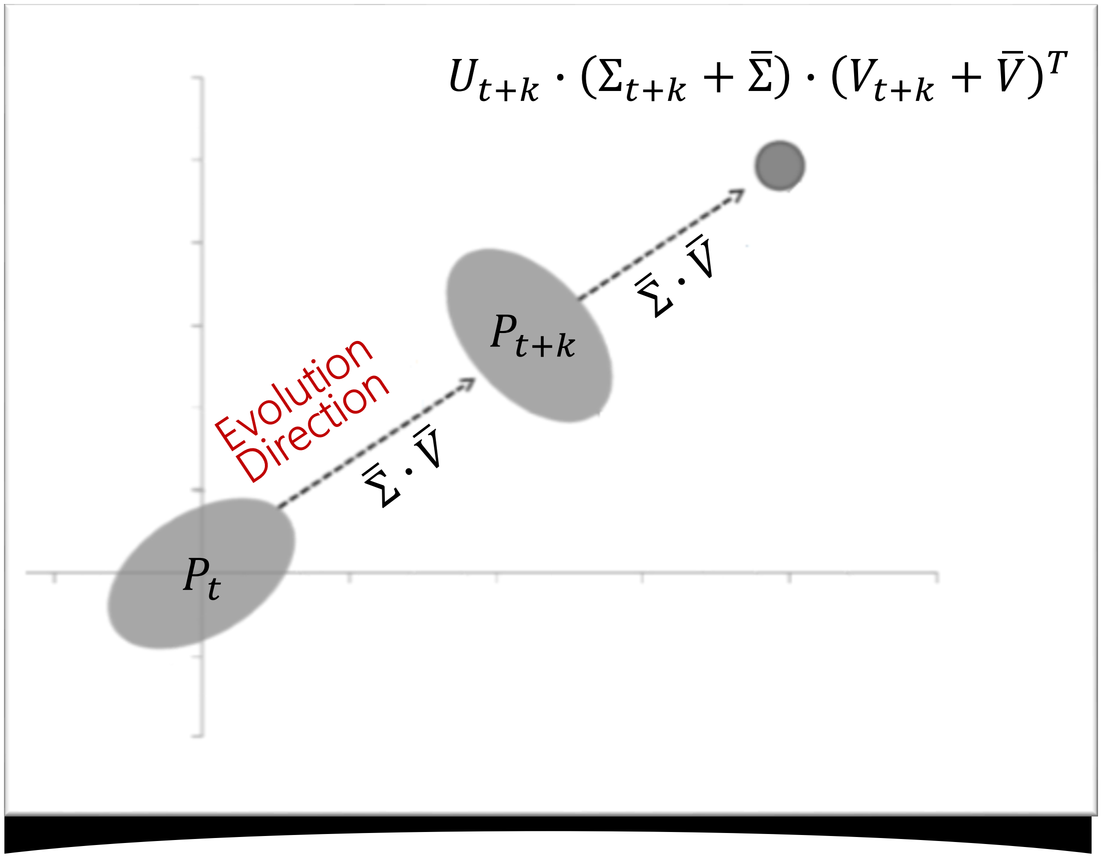
$$\overline{V} = V_{t+k} - V_t$$

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

Using SVD for Evolution Direction

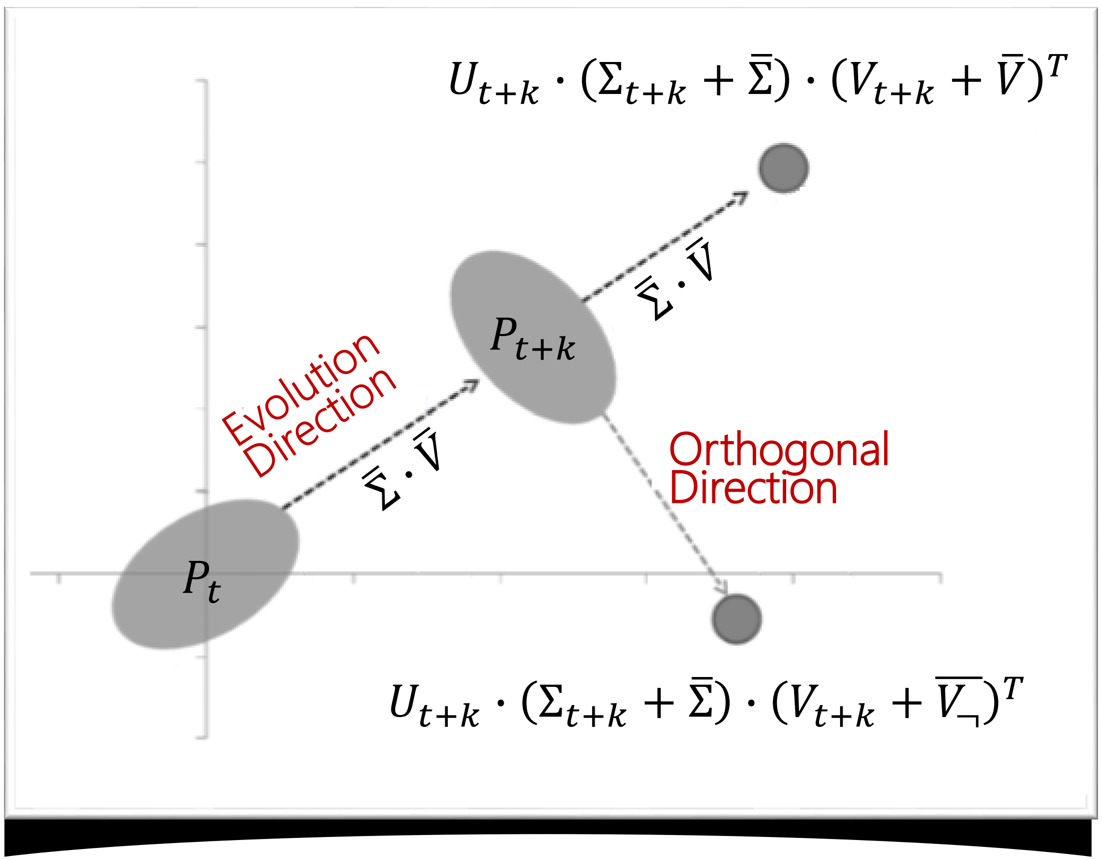


Using SVD for Evolution Direction

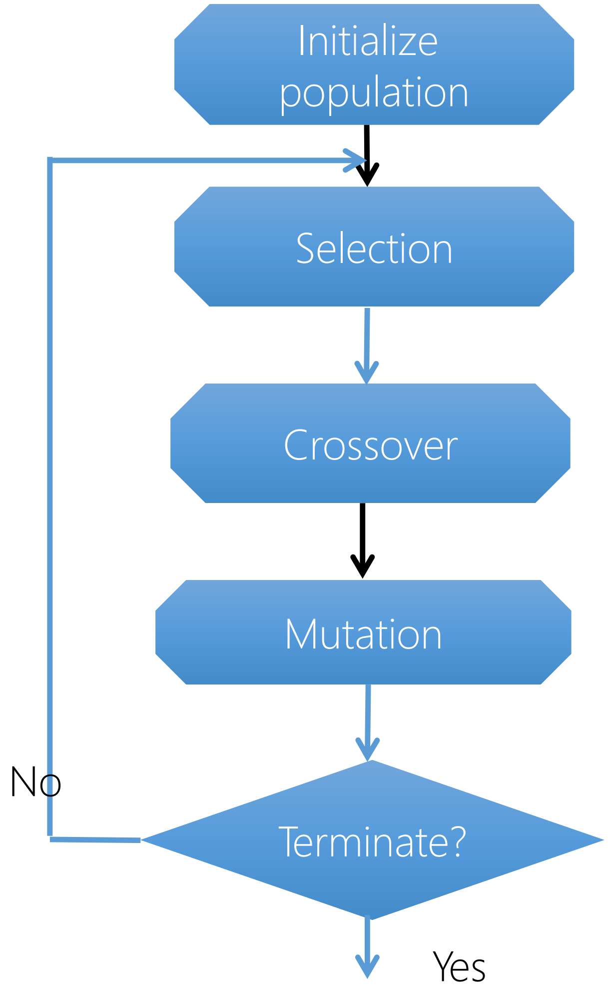


Using SVD for Evolution Direction

Then, we construct a new orthogonal population as follows

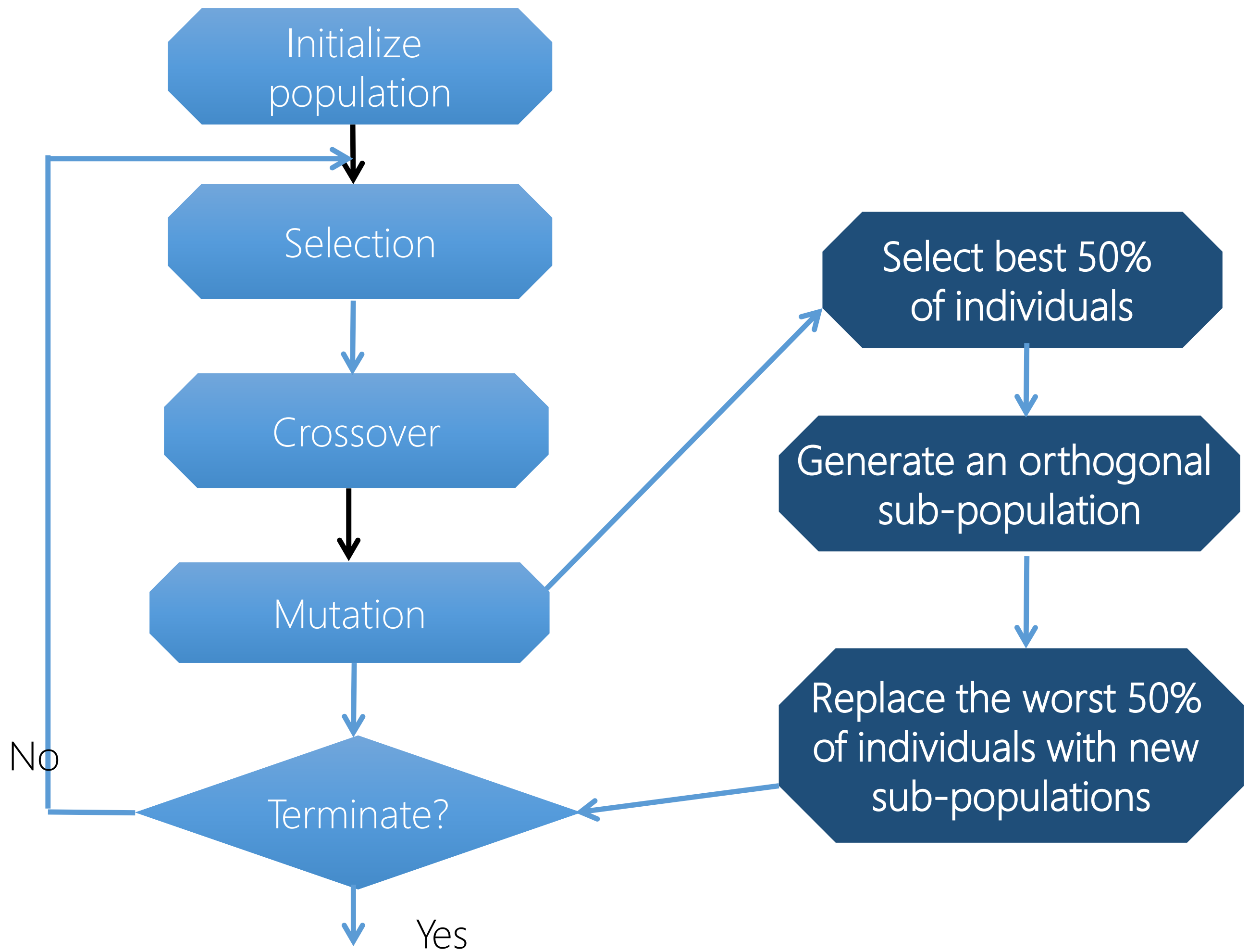


Integration SVD with Standard GA



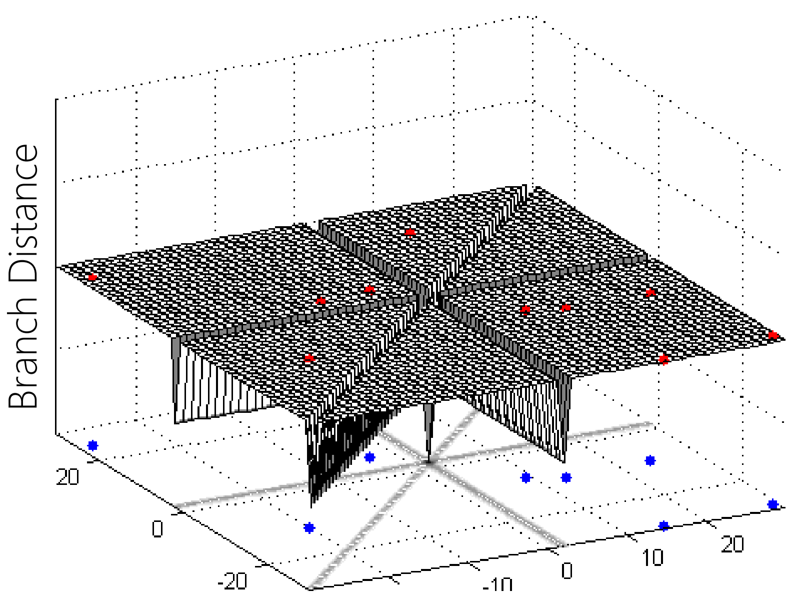
- Rank Scaling Selection
- Single-point crossover
- Uniform mutation

SVD + Standard GAs

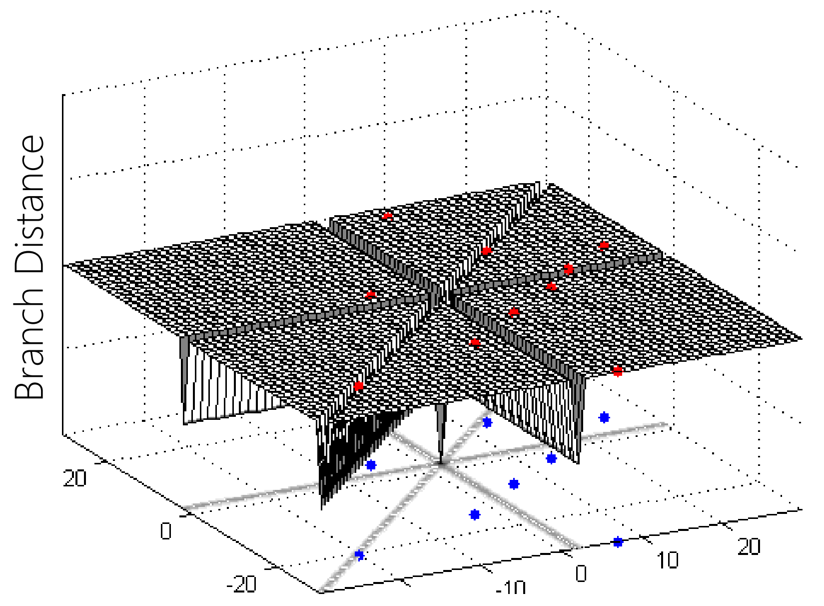


Simulation on Triangle Program

Standard GA



SVD-GA



Empirical study

No.	Name	Coverage Goals
P1	ArithmeticUtils	99
P2	Arrays	75
P3	Beta	90
P4	CreditCardValidator	32
P5	Complex	126
P6	FastMath	60
P7	Fraction	108
P8	IPAddressValidator	243
P9	LUDecomposition	76
P10	KolmogorovDistribution	50
P11	QRDecomposition	72
P12	Quadratic	7
P13	RootsOfUnity	27
P14	SaddlePointExansion	16
P15	Sort	70
P16	Tomorrow	107
P17	TriangularDistribution	50

Experimented Algorithms:

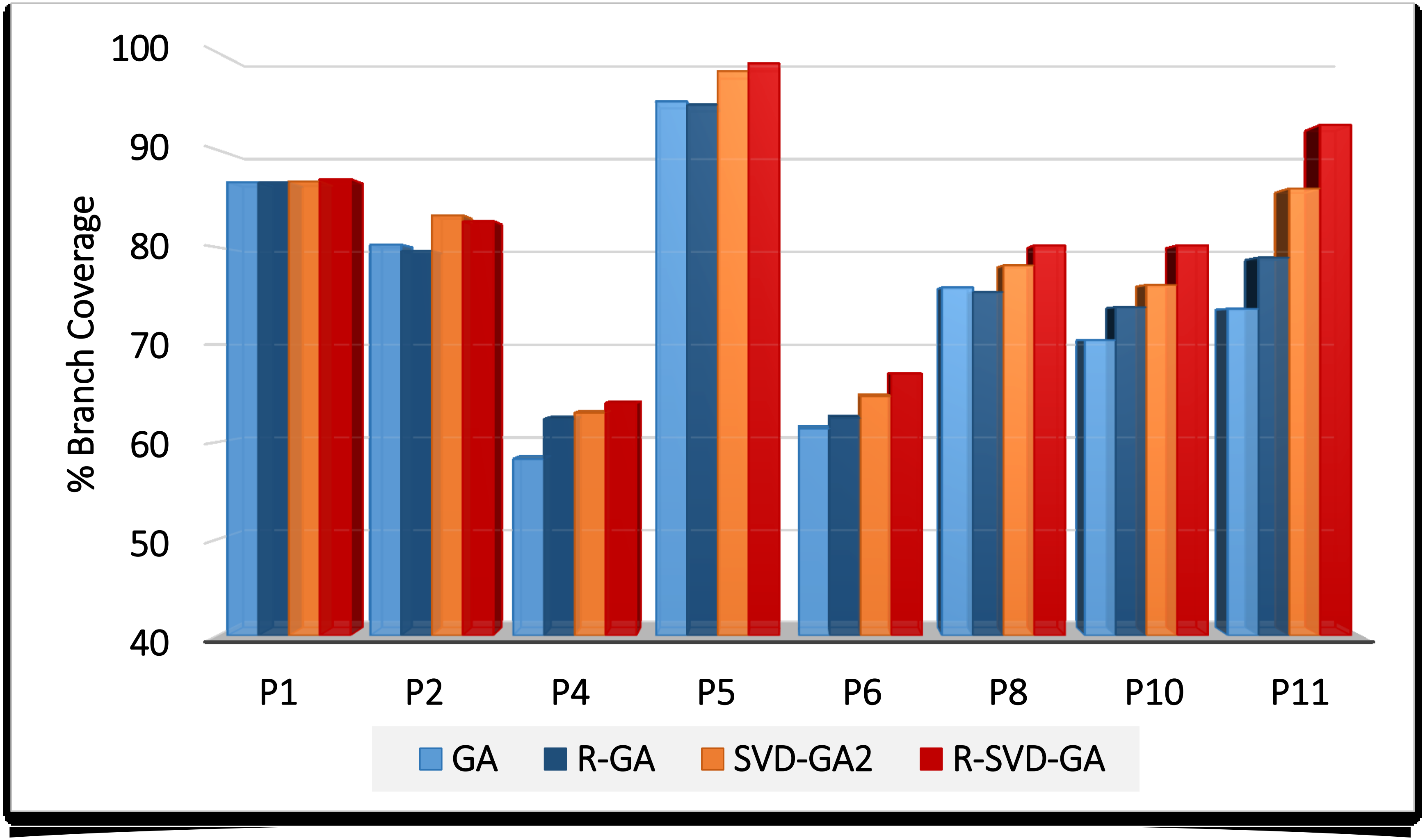
1. SVD-GA
2. R-GA
3. R-SVD-GA
4. Standard GA

Performance metrics:

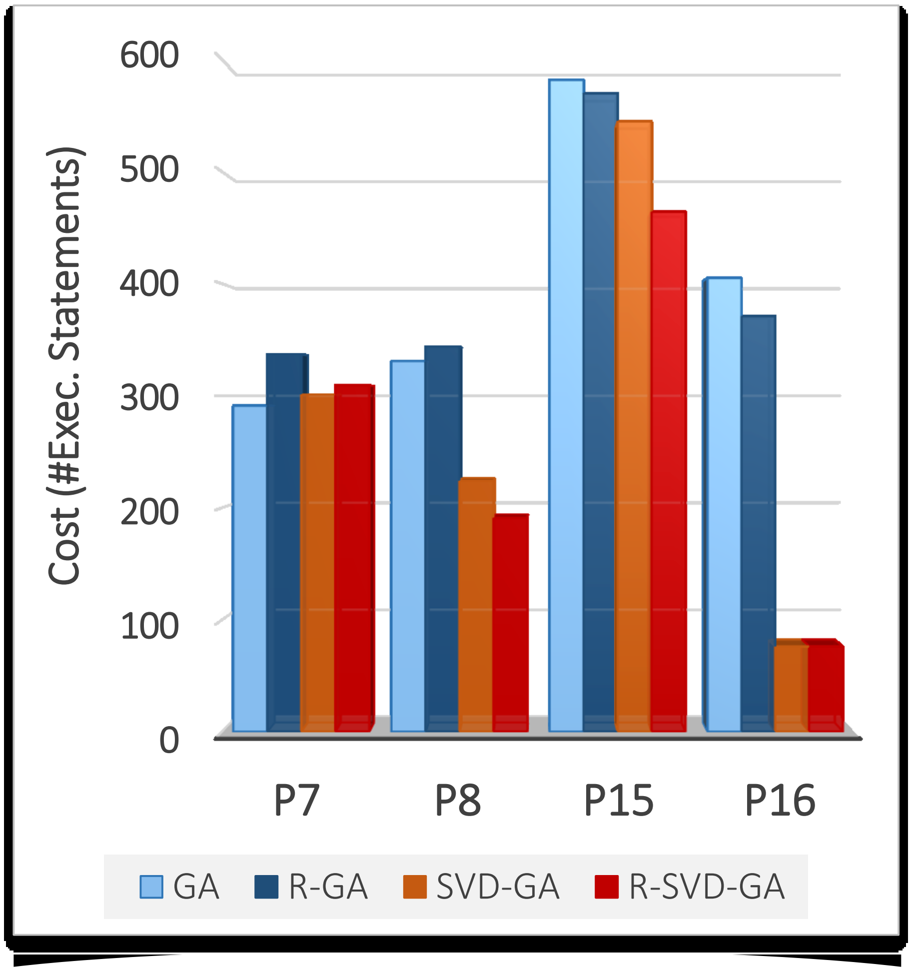
Effectiveness = % covered braches

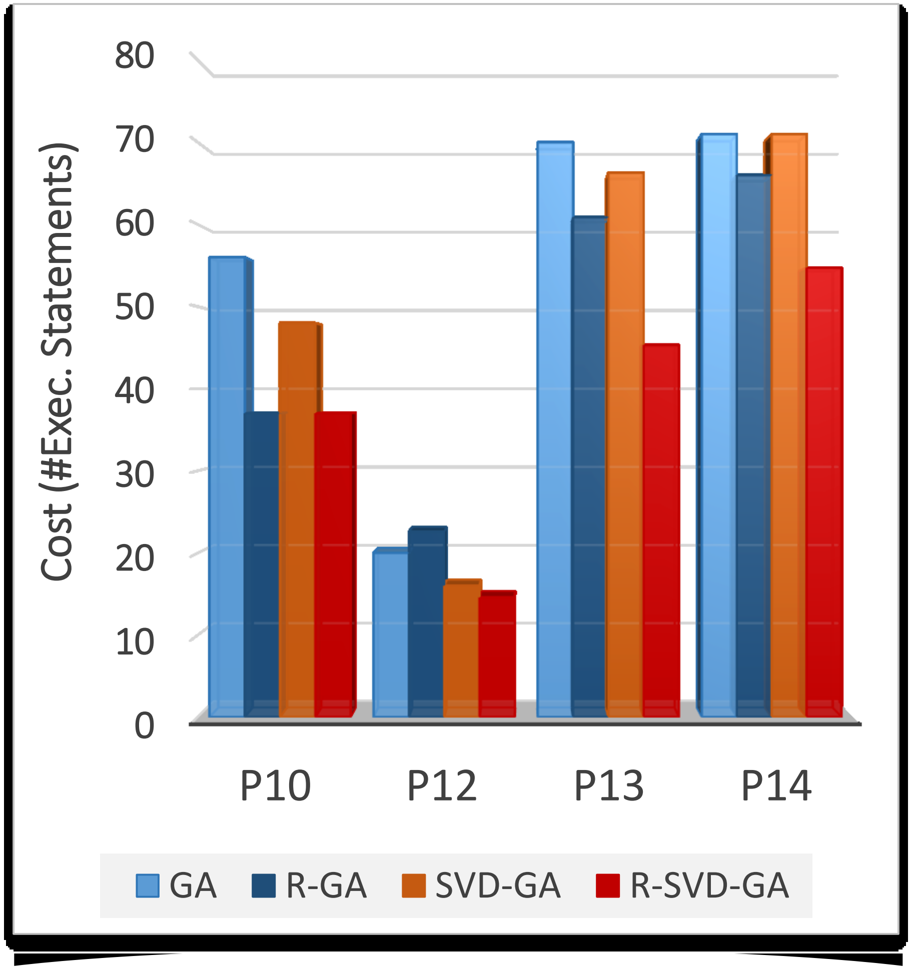
Efficiency/cost = # executed statements

RQ1: Does orthogonal exploration improve the effectiveness of evolutionary test case generation?

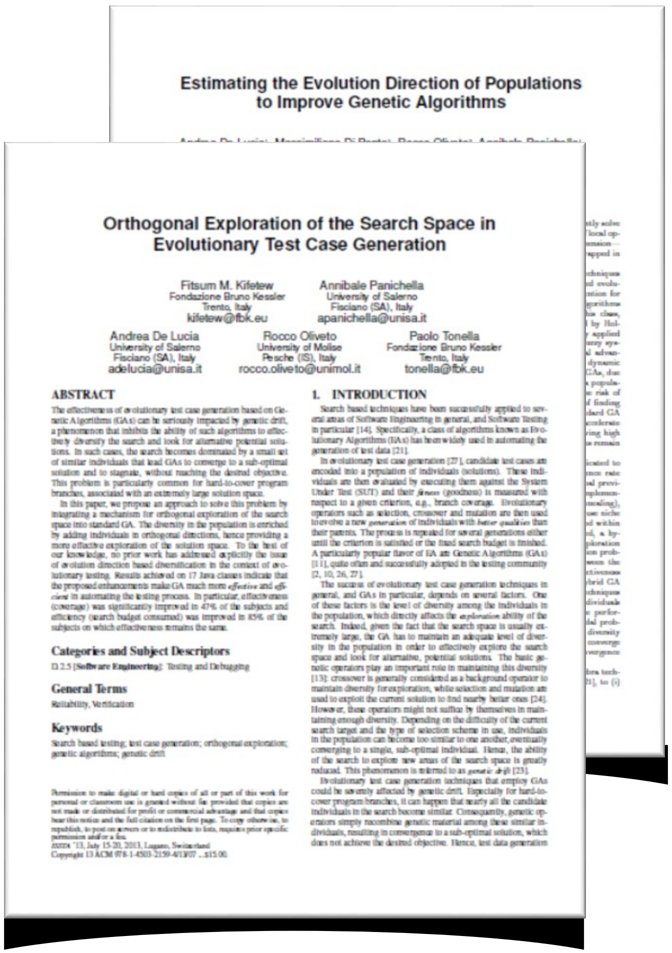


RQ2: Does orthogonal exploration improve the efficiency of evolutionary test case generation?





Orthogonal exploration


Estimating the Evolution Direction of Populations to Improve Genetic Algorithms

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**

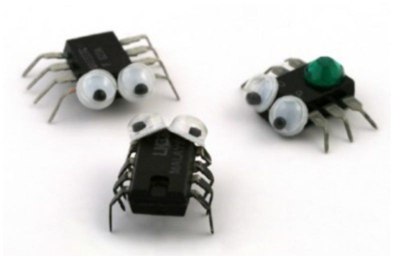
GECCO 2012

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

Main Contributions

Search-Based Program Comprehension



Multi-Objectives Defect Prediction

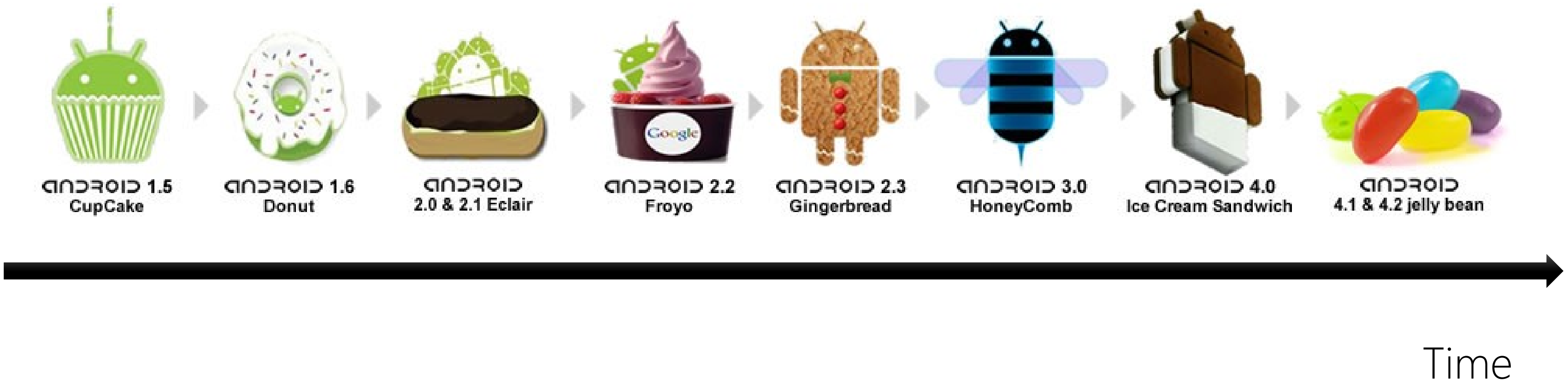
Search-Based Test Data Generation

Multi-Objective Test Suite Optimization

Software Evolution

Software continuously changes (evolves):

- Add new functionalities
- Removing old functionalities
- Bug fixing activities
- …



Regression Testing

Software before changes

- Test Case 1
- Test Case 2
- Test Case 3
- …
- Test Case n

Software after changes

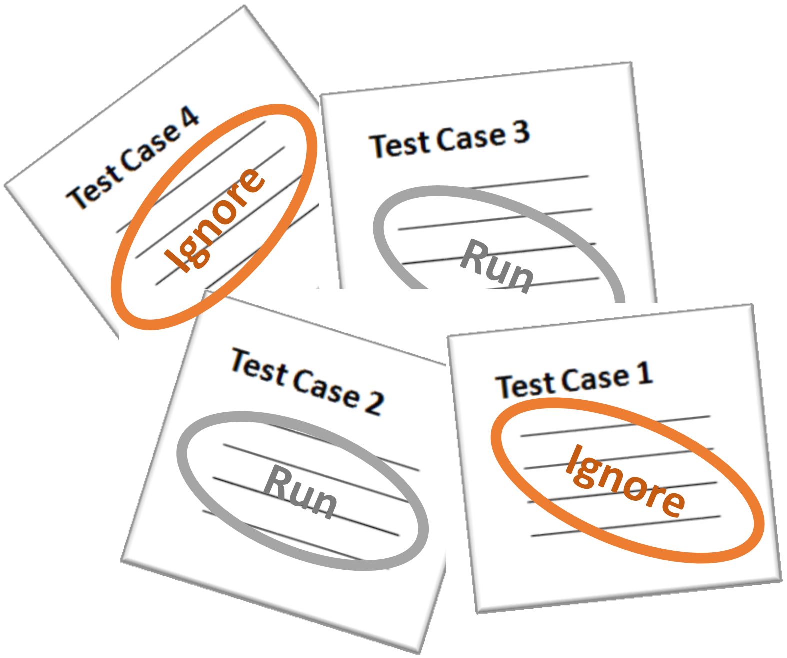
- Test Case 1
- Test Case 2
- Test Case 3
- …
- Test Case n

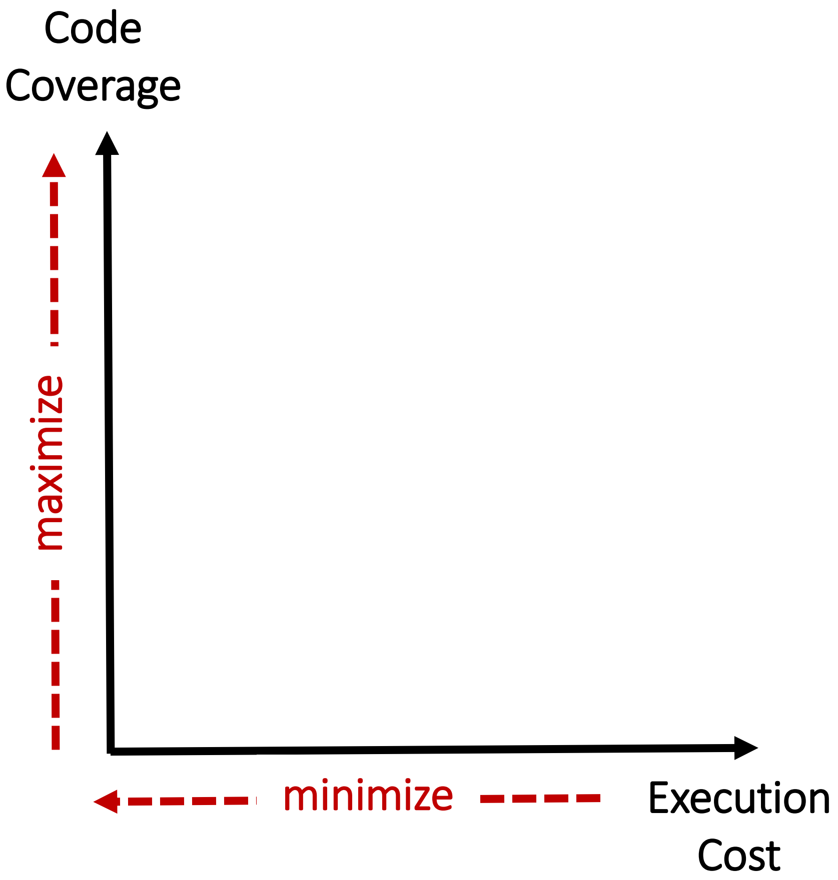
Regression Testing is time consuming

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*. TSE 2010

Test Suite Optimization

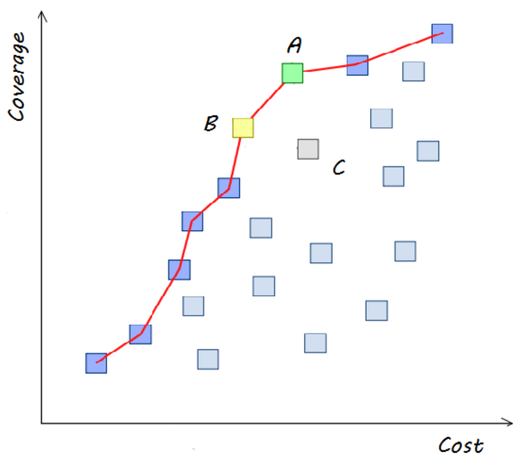




Multi-Criteria Regression Testing

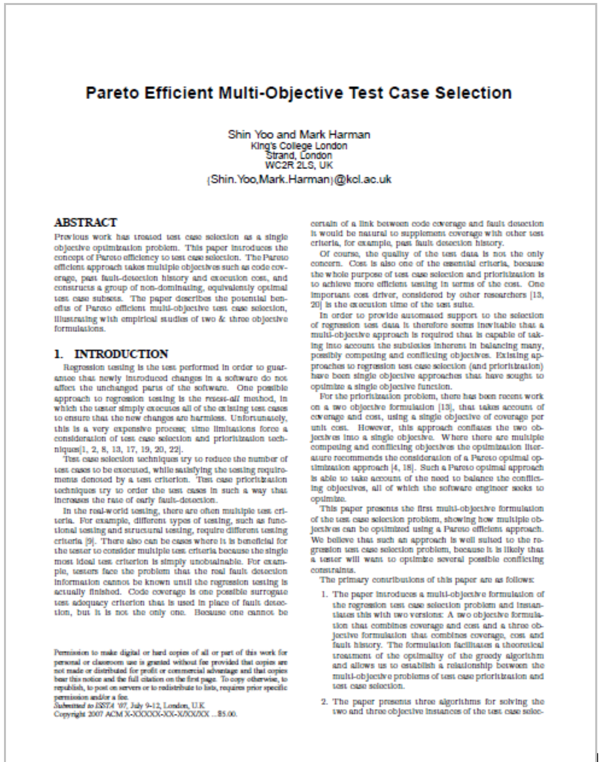
Multiple objectives are optimized using Pareto efficient approaches

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:

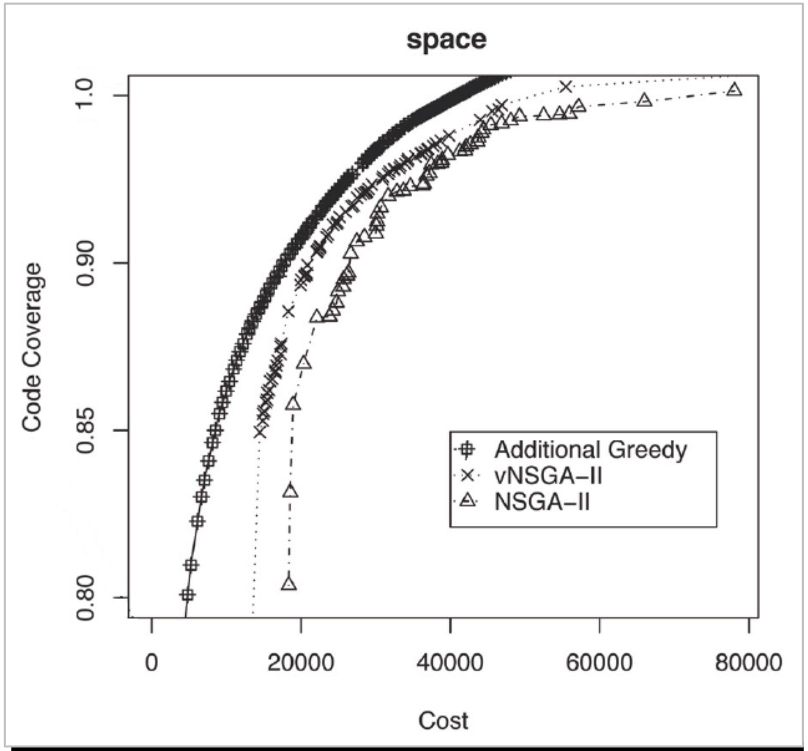
1. The paper introduces a multi-objective formulation of the regression test case selection problem and instantiates this with two versions: A two objective formulation that combines coverage and cost and a three objective formulation that combines coverage, cost and fault history. The formulation facilitates 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.

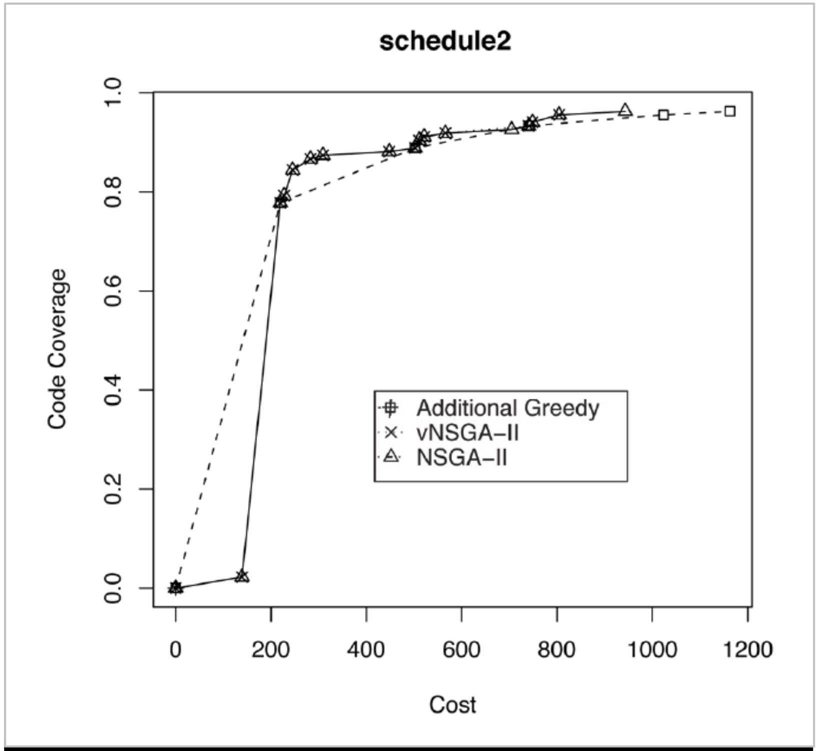
2. The paper presents three algorithms for solving the two and three objective instances of the test case selec-

Permission to make digital or hard copies of all or part 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 bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Submitted to ISSTA '07, July 9-12, London, U.K.
Copyright 2007 ACM X-XXXXX-XX-X/XX/XX ...$5.00.

Multi-Criteria Regression Testing

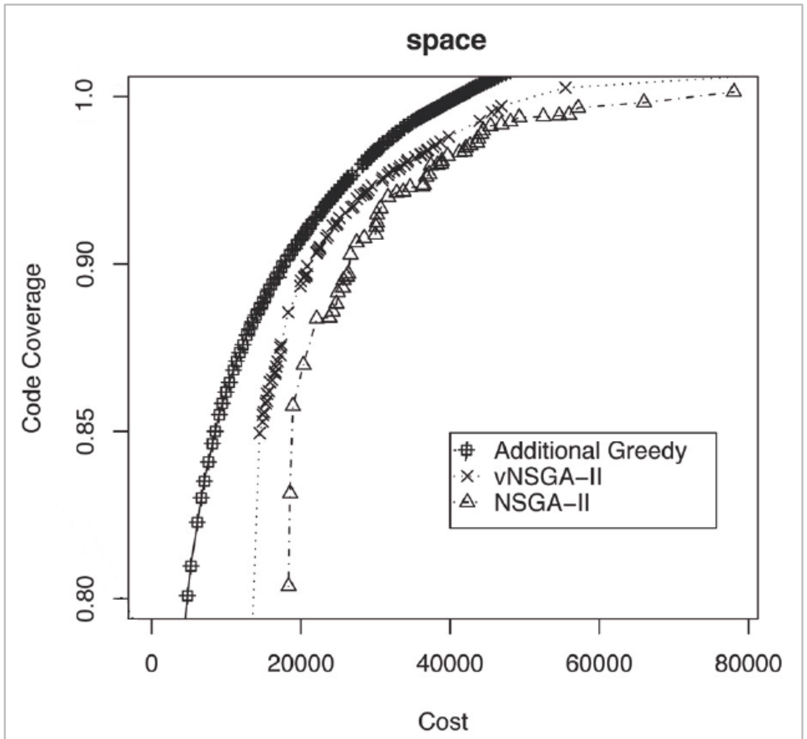
There is no clear winner



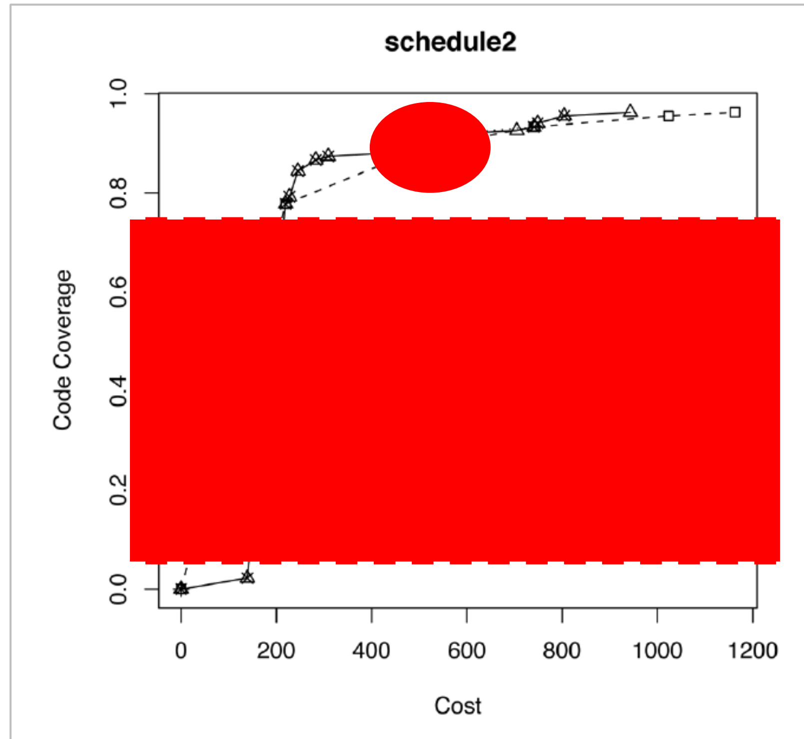


Multi-Criteria Regression Testing

There is no clear winner



Population Drift

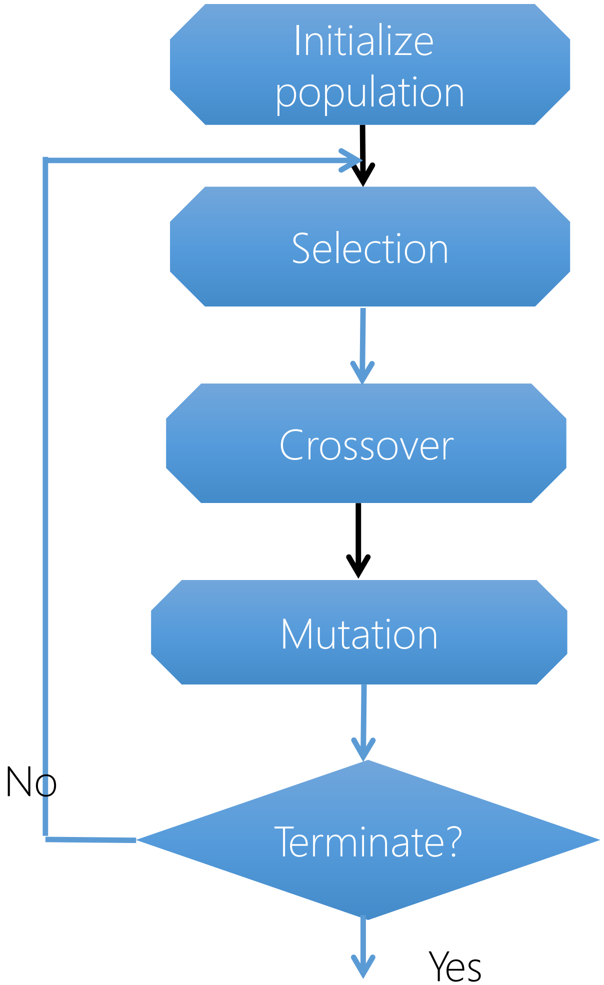


Multi-Criteria Regression Testing

There is no clear winner

Can we do better?

Diversity Injection in NSGA-II

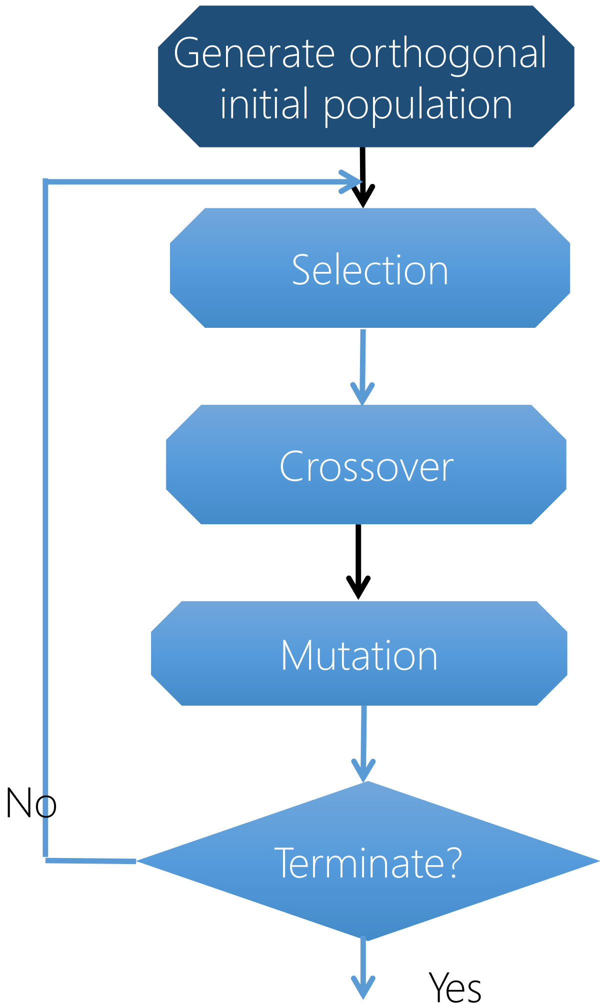


- Non Dominated Sorting Algorithm
- Crowding Distance
- Tournament Selection

- Multi-points crossover

- Bit-flip mutation

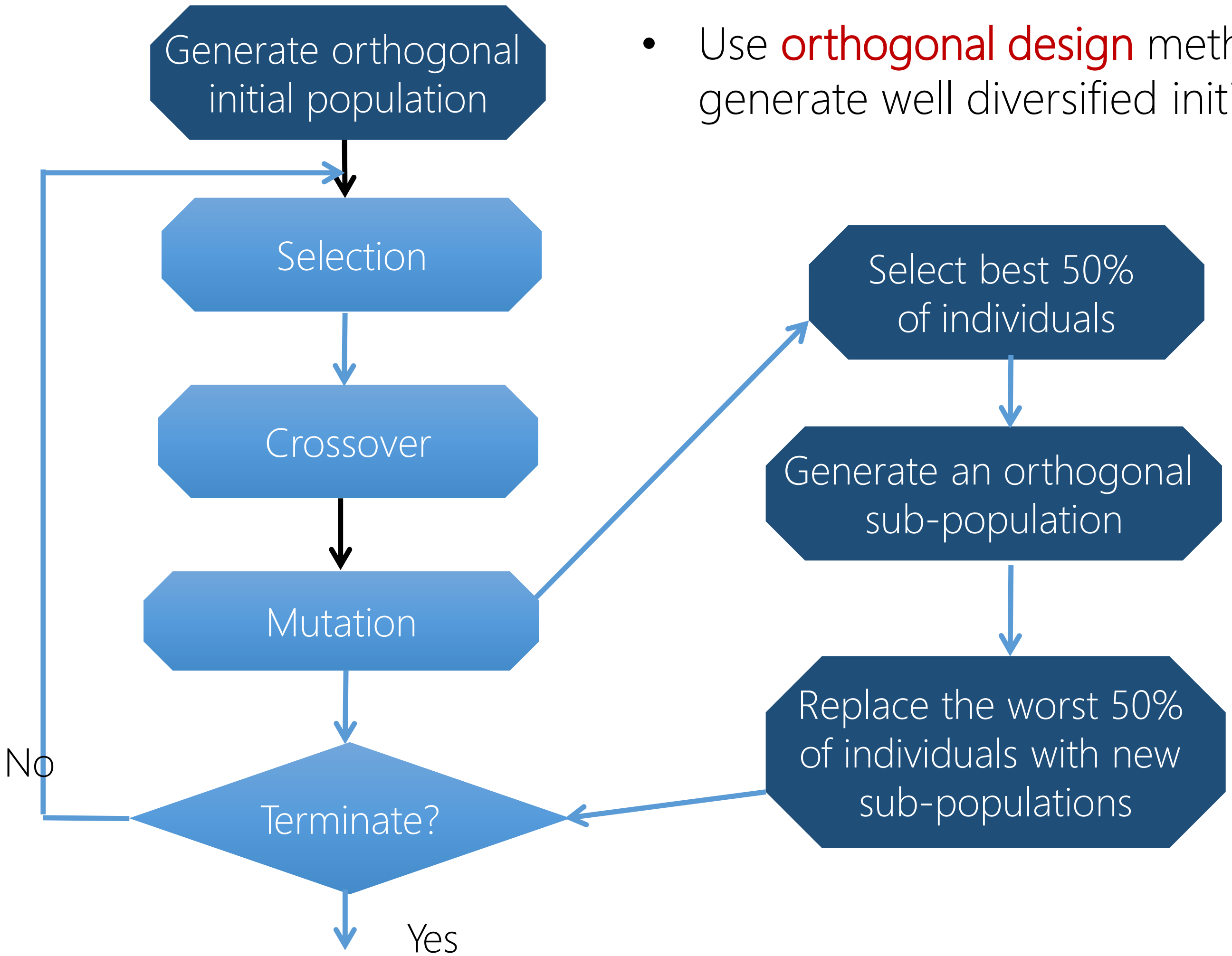
Diversity Injection in NSGA-II



- Use orthogonal design methodology to generate well diversified initial population

SVD + NSGA-II

- Use orthogonal design methodology to generate well diversified initial population



Empirical Evaluation

Software systems:

No.	Name	LOC	Test Suite Size
1	bash	59,846	1,200
2	flex	10,459	567
3	grep	10,068	808
4	gzip	5,680	215
5	printtokens	726	4,130
6	printtokens2	520	4,115
7	schedule	412	2,650
8	sechedule2	374	2,710
9	sed	14,427	360
10	space	6,199	13,583
11	vim	122,169	975

Experimented Algorithms:

1. SVD-NSGA-II + Init. Pop
2. NSGA-II
3. Additional Greedy Algorithm

Problems:

1. 2-objectives
   - Execution Cost
   - Code Coverage
2. 3-objectives
   - 2-objectives + Past Faults Coverage

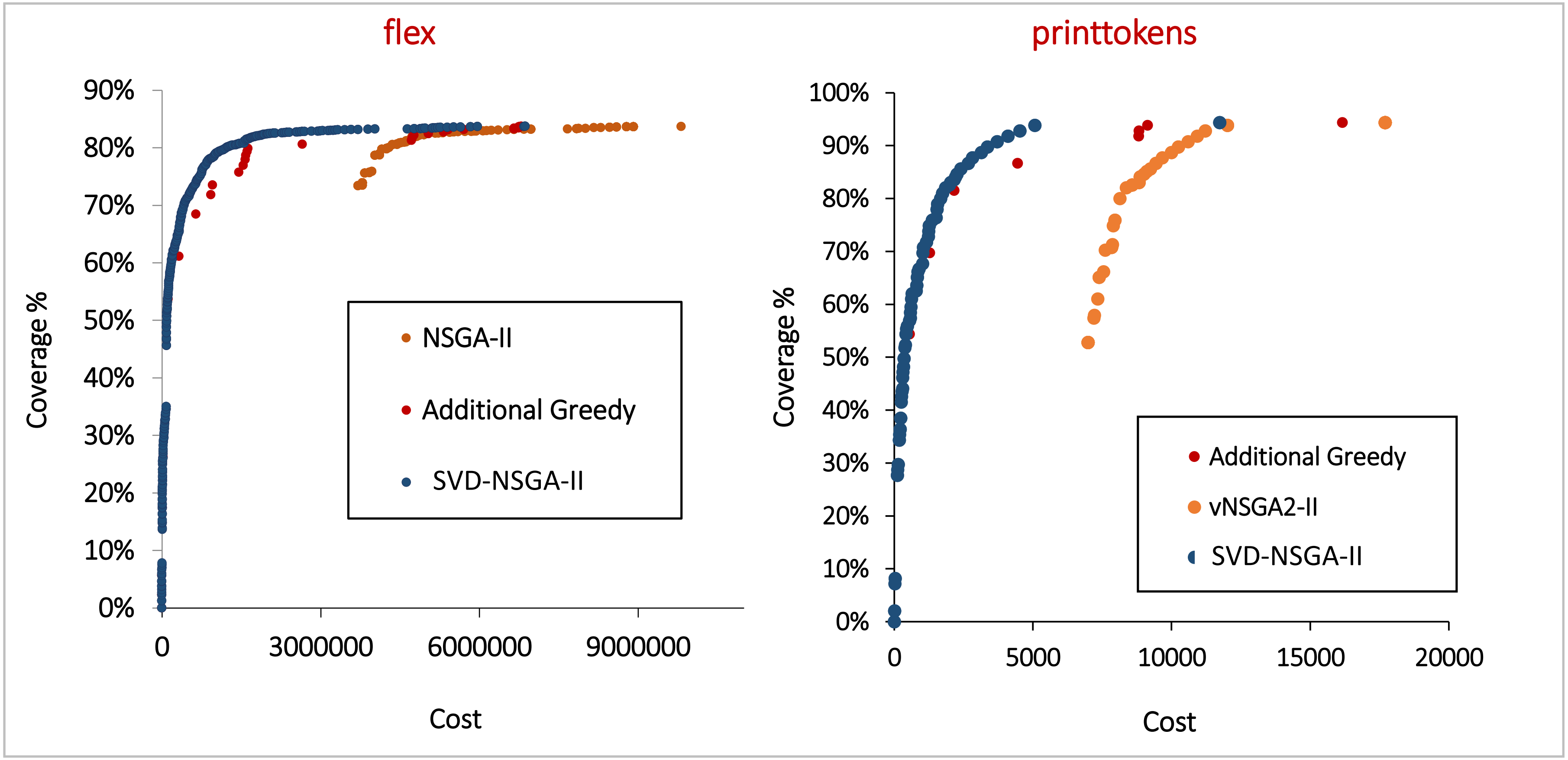
Performance metrics:

Pareto optimal solutions

% hypervolume = % detected faults per unit time

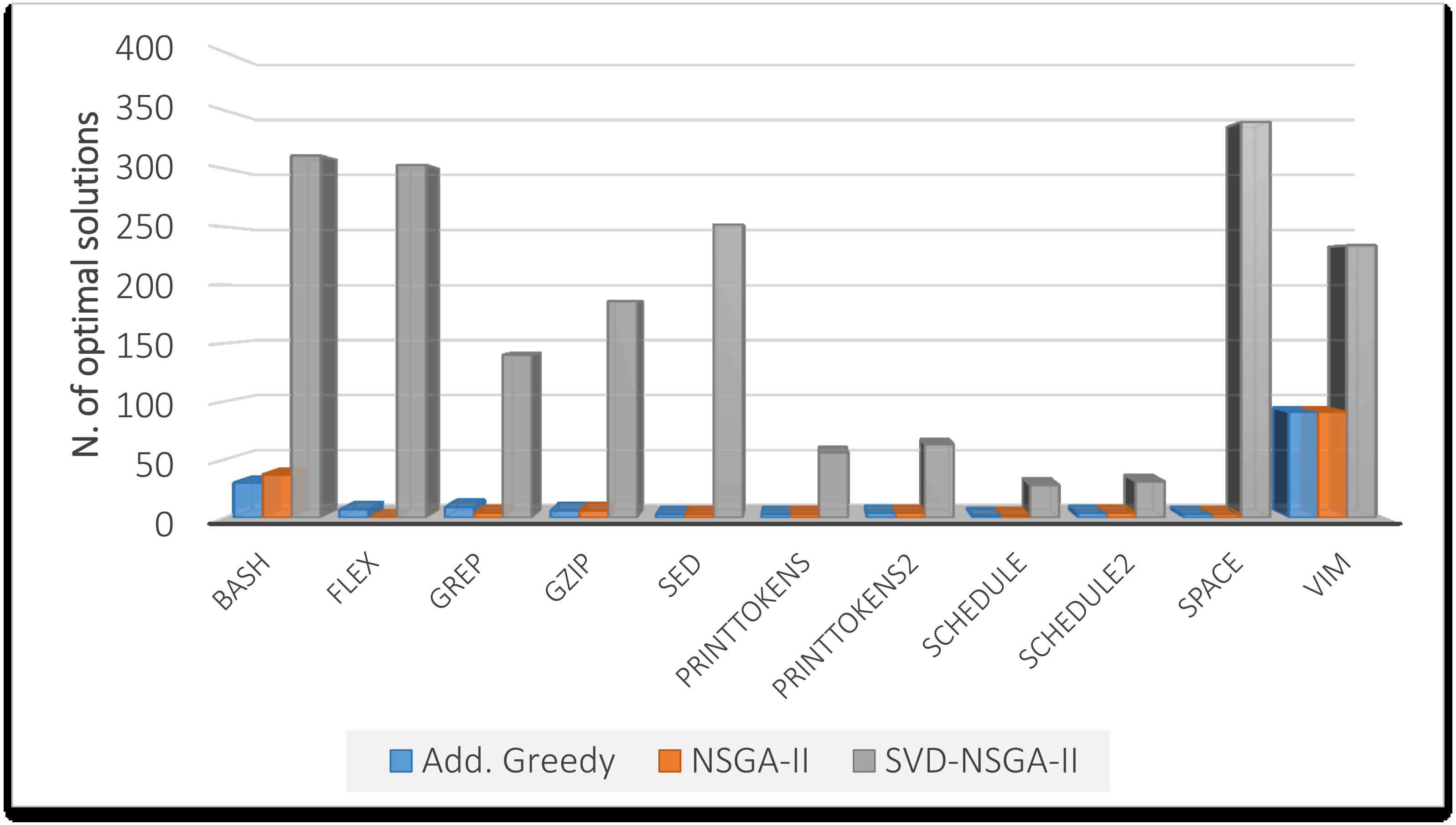
Results

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



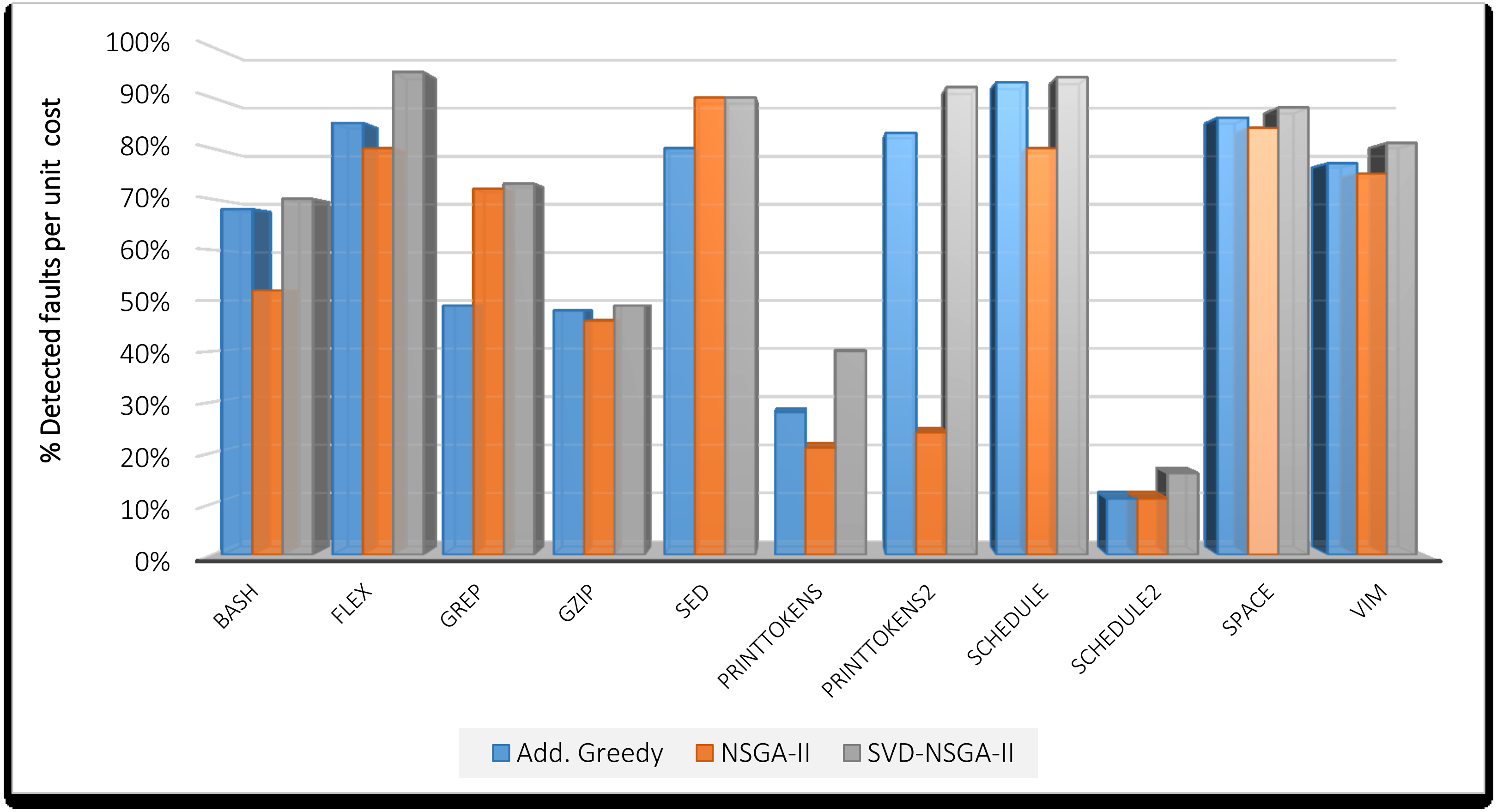
Results

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

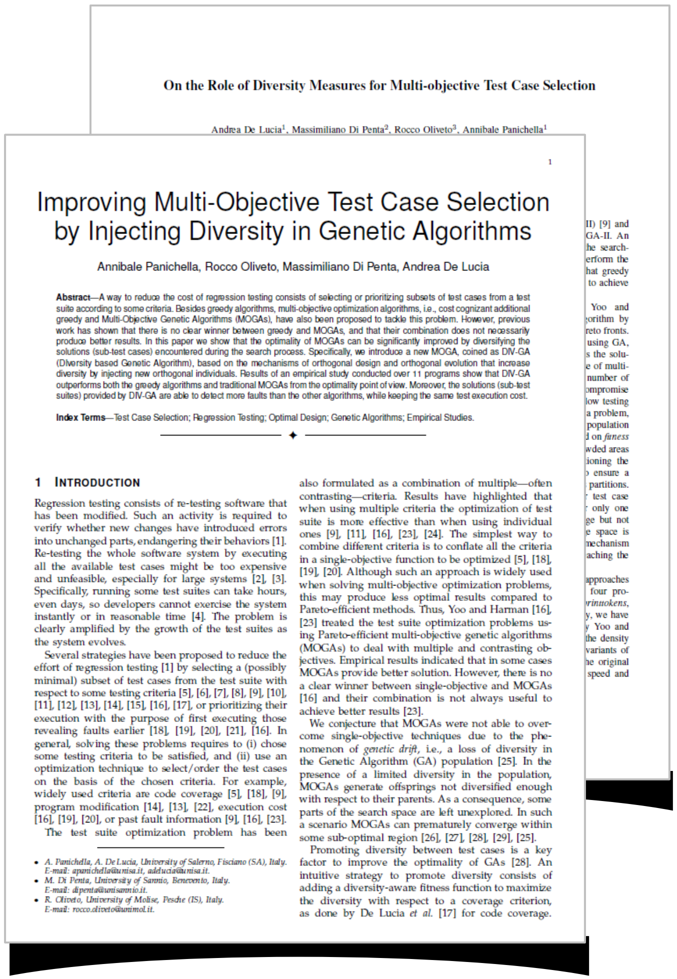


Results

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



Diversity in T.S. Optimization


On the Role of Diversity Measures for Multi-objective Test Case Selection

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

1

Improving Multi-Objective Test Case Selection by Injecting Diversity in Genetic Algorithms

Annibale Panichella, Rocco Oliveto, Massimiliano Di Penta, Andrea De Lucia

Abstract—A way to reduce the cost of regression testing consists of selecting or prioritizing subsets of test cases from a test suite according to some criteria. Besides greedy algorithms, multi-objective optimization algorithms, i.e., cost cognizant additional greedy and Multi-Objective Genetic Algorithms (MOGAs), have also been proposed to tackle this problem. However, previous work has shown that there is no clear winner between greedy and MOGAs, and that their combination does not necessarily produce better results. In this paper we show that the optimality of MOGAs can be significantly improved by diversifying the solutions (sub-test cases) encountered during the search process. Specifically, we introduce a new MOGA, coined as DIV-GA (DIversity based Genetic Algorithm), based on the mechanisms of orthogonal design and orthogonal evolution that increase diversity by injecting new orthogonal individuals. Results of an empirical study conducted over 11 programs show that DIV-GA outperforms both the greedy algorithms and traditional MOGAs from the optimality point of view. Moreover, the solutions (sub-test suites) provided by DIV-GA are able to detect more faults than the other algorithms, while keeping the same test execution cost.

Index Terms—Test Case Selection; Regression Testing; Optimal Design; Genetic Algorithms; Empirical Studies.

1 Introduction

Regression testing consists of re-testing software that has been modified. Such an activity is required to verify whether new changes have introduced errors into unchanged parts, endangering their behaviors [1]. Re-testing the whole software system by executing all the available test cases might be too expensive and unfeasible, especially for large systems [2], [3]. Specifically, running some test suites can take hours, even days, so developers cannot exercise the system instantly or in reasonable time [4]. The problem is clearly amplified by the growth of the test suites as the system evolves.

Several strategies have been proposed to reduce the effort of regression testing [1] by selecting a (possibly minimal) subset of test cases from the test suite with respect to some testing criteria [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], or prioritizing their execution with the purpose of first executing those revealing faults earlier [18], [19], [20], [21], [16]. In general, solving these problems requires to (i) chose some testing criteria to be satisfied, and (ii) use an optimization technique to select/order the test cases on the basis of the chosen criteria. For example, widely used criteria are code coverage [5], [18], [9], program modification [14], [13], [22], execution cost [16], [19], [20], or past fault information [9], [16], [23].

The test suite optimization problem has been also formulated as a combination of multiple—often contrasting—criteria. Results have highlighted that when using multiple criteria the optimization of test suite is more effective than when using individual ones [9], [11], [16], [23], [24]. The simplest way to combine different criteria is to conflate all the criteria in a single-objective function to be optimized [5], [18], [19], [20]. Although such an approach is widely used when solving multi-objective optimization problems, this may produce less optimal results compared to Pareto-efficient methods. Thus, Yoo and Harman [16], [23] treated the test suite optimization problems using Pareto-efficient multi-objective genetic algorithms (MOGAs) to deal with multiple and contrasting objectives. Empirical results indicated that in some cases MOGAs provide better solution. However, there is no a clear winner between single-objective and MOGAs [16] and their combination is not always useful to achieve better results [23].

We conjecture that MOGAs were not able to overcome single-objective techniques due to the phenomenon of *genetic drift*, i.e., a loss of diversity in the Genetic Algorithm (GA) population [25]. In the presence of a limited diversity in the population, MOGAs generate offsprings not diversified enough with respect to their parents. As a consequence, some parts of the search space are left unexplored. In such a scenario MOGAs can prematurely converge within some sub-optimal region [26], [27], [28], [29], [25].

Promoting diversity between test cases is a key factor to improve the optimality of GAs [28]. An intuitive strategy to promote diversity consists of adding a diversity-aware fitness function to maximize the diversity with respect to a coverage criterion, as done by De Lucia *et al.* [17] for code coverage.

- *A. Panichella, A. De Lucia, University of Salerno, Fisciano (SA), Italy. E-mail: apanichella@unisa.it, adelucia@unisa.it.*
- *M. Di Penta, University of Sannio, Benevento, Italy. E-mail: dipenta@unisannio.it.*
- *R. Oliveto, University of Molise, Pesche (IS), Italy. E-mail: rocco.oliveto@unimol.it.*

On the role of diversity measures for multi-objective test case selection A. De Lucia, M. Di Penta, R. Oliveto, <u>A. Panichella</u>. International Workshop on Automation of Software Test (AST) 2012

Improving Multi-Objective Search Based Test Suite Optimization through Diversity Injection. <u>A. Panichella</u>, R. Oliveto, M. Di Penta, A. De Lucia. *In major revision* at IEEE Transactions on Software Engineering (TSE).

Summary

Search-Based Program Comprehension

Multi-Objectives Defect Prediction

Search-Based Test Data Generation

Multi-Objective Test Suite Optimization

Thanks!

Question?