



A Revised Classification of Product Sampling for Software Product Lines

Sabrina Böhm, Aaron Molt, Tim Jannik Schmidt, Thomas Thüm | March 23 - March 27 2026



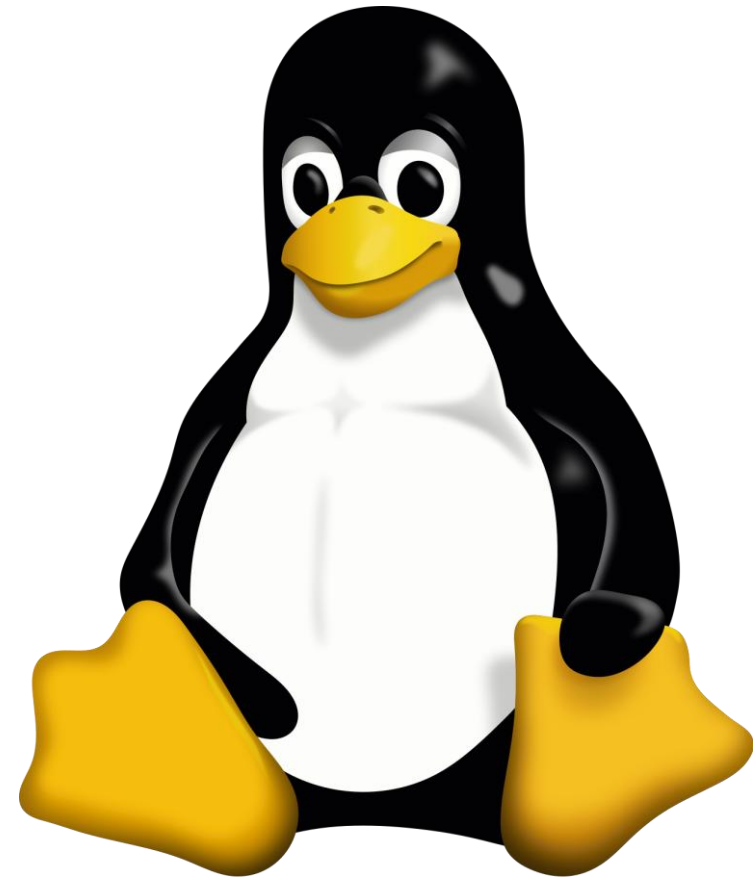
universität
uulm

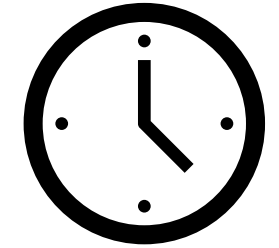
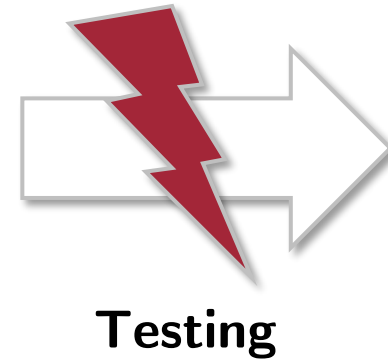
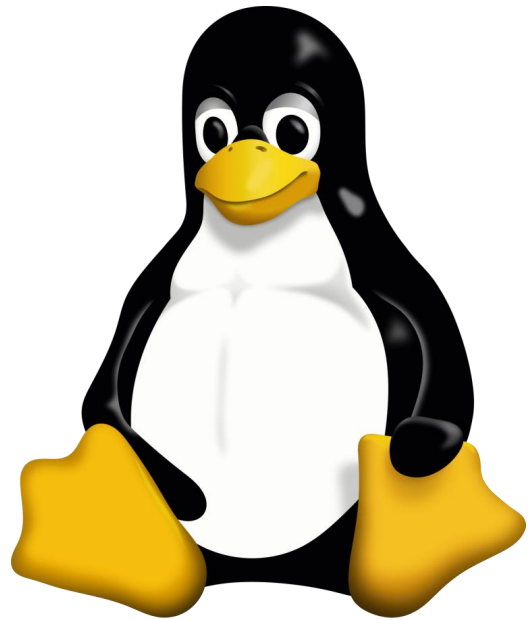


Product Lines



(Software) Product Lines





```
> testsuite@1.0.0 test
PASS ./index.test.js
my tests
  ✓ are perfect
  ✓ always work

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Time:        0.513 s, estimated 1s
Ran all test suites.
```

To Variability! JHipster: A Playground for Web-Apps Analyses

Axel Hallin
Philippe Desrosiers
University of Namur, Belgium
axel.hallin@unam.ac.be

Alexandre Nuttack
Philippe Desrosiers
University of Namur, Belgium
alexandre.nuttack@unam.ac.be

Mathieu Acher
IRISA, University of Rennes 1,
France
mathieu.acher@irisa.fr

Xavier Devrey
Philippe Desrosiers
University of Namur, Belgium
xavier.devrey@unam.ac.be

Gilles Perrouin
Philippe Desrosiers
University of Namur, Belgium
gilles.perrouin@unam.ac.be

Patrick Heymans
Philippe Desrosiers
University of Namur, Belgium
patrick.heyman@unam.ac.be

Keywords: Code Smells, WebApps, Variability-related Analysis

1. INTRODUCTION
JHipster [2] is an open-source generator for Web applications. It generates code for a wide range of technologies, including Java, JavaScript, CSS, and HTML. It is designed to be used as a starting point for developing new web applications, and it provides a wide range of features and options to customize the generated code.

July 27, 2014. 16:17. WSPC/INSTRUCTION FILE jhipster

Proceeding Paper
© World Scientific Publishing Company

EXPERIMENTS ON OPTIMIZING THE PERFORMANCE OF THE JHIPSTER GENERATOR

ALEXANDRE NUTTACK¹ AND BERTRAND KIEHN² AND CHRISTIAN BARDIOL³ AND NICOLAS BILLOREAU⁴ AND XAVIER DEVREY⁵
¹ IRISA, University of Rennes 1, France
² IRISA, University of Rennes 1, France
³ IRISA, University of Rennes 1, France
⁴ IRISA, University of Rennes 1, France
⁵ IRISA, University of Rennes 1, France

Received April 2014
Revised July 2014
Accepted July 2014
Communicated by Gene Golub

ABSTRACT
A detailed technique for automatically solving integer partial differential equations is presented. This technique is based on a combination of symbolic and numerical methods. It is applied to the problem of finding the shortest path in a graph. The results show that this technique is more efficient than other methods.

KEYWORDS: Integer Partial Differential Equations, Symbolic and Numerical Methods, Shortest Path Problem

1. INTRODUCTION
The problem of finding the shortest path in a graph is a classic problem in computer science. It has many applications, such as in network routing and in the design of integrated circuits. This paper presents a new algorithm for solving this problem. The algorithm is based on a combination of symbolic and numerical methods. It is more efficient than other methods.

2. PRELIMINARIES
Let $G = (V, E)$ be a graph with n vertices and m edges. Let s and t be two vertices in V . The shortest path from s to t is the path with the minimum number of edges. This paper presents a new algorithm for finding this path. The algorithm is based on a combination of symbolic and numerical methods.

3. THE ALGORITHM
The algorithm consists of two main steps. In the first step, the graph is converted into a matrix. In the second step, the matrix is processed to find the shortest path. The algorithm is more efficient than other methods.

4. EXPERIMENTAL RESULTS
The algorithm was tested on a set of graphs. The results show that it is more efficient than other methods. It is able to find the shortest path in a graph with a large number of vertices and edges.

5. CONCLUSION
This paper presents a new algorithm for finding the shortest path in a graph. The algorithm is based on a combination of symbolic and numerical methods. It is more efficient than other methods.

REFERENCES
[1] Dijkstra, E. W. A note on algorithms for solving the shortest path problem. *Mathematische Annalen*, 1959, 159, 371-375.
[2] JHipster, an open-source generator for Web applications. <http://jhipster.github.io/>

CONTACT: Alexandre Nuttack, alexandre.nuttack@unam.ac.be

Cost-Efficient Sampling for Performance of Configurable Systems

Almida Sarkar, James Guo, Nohbert Siegmund, Sven Apel
University of Waterloo, Canada
Email: asarkar@uwaterloo.ca, Email: nohbert.siegmund@uwaterloo.ca

Abstract: A key challenge of the development and maintenance of configurable systems is the performance of product variants. It is difficult to maintain the performance of product variants, due to feature constraints. Previous approaches predict performance based on small number of measured variants, but it is still open how to dynamically determine an ideal number of feature configurations to measure. In this paper, we adopt the widely-used sampling strategy for performance prediction and measurement effort. We propose a new cost-efficient sampling strategy, which considers prediction accuracy and measurement effort simultaneously. We compare our approach with a traditional method based on small number of measured variants. Our approach can reduce the number of feature configurations to measure and provide guidance for feature selection to predict performance by sampling.

Keywords: Feature Configuration, Performance Prediction, Sampling

1. INTRODUCTION
Sampling is a common approach for performance prediction of configurable systems. It involves selecting a small number of feature configurations to measure, and using the results to predict the performance of all possible configurations. This approach is cost-efficient, but it is still open how to dynamically determine an ideal number of feature configurations to measure. In this paper, we propose a new cost-efficient sampling strategy, which considers prediction accuracy and measurement effort simultaneously. We compare our approach with a traditional method based on small number of measured variants. Our approach can reduce the number of feature configurations to measure and provide guidance for feature selection to predict performance by sampling.

July 27, 2014. 16:17. WSPC/INSTRUCTION FILE jhipster

Proceeding Paper
© World Scientific Publishing Company

EXPERIMENTS ON OPTIMIZING THE PERFORMANCE OF THE JHIPSTER GENERATOR

ALEXANDRE NUTTACK¹ AND BERTRAND KIEHN² AND CHRISTIAN BARDIOL³ AND NICOLAS BILLOREAU⁴ AND XAVIER DEVREY⁵
¹ IRISA, University of Rennes 1, France
² IRISA, University of Rennes 1, France
³ IRISA, University of Rennes 1, France
⁴ IRISA, University of Rennes 1, France
⁵ IRISA, University of Rennes 1, France

Received April 2014
Revised July 2014
Accepted July 2014
Communicated by Gene Golub

ABSTRACT
A detailed technique for automatically solving integer partial differential equations is presented. This technique is based on a combination of symbolic and numerical methods. It is applied to the problem of finding the shortest path in a graph. The results show that this technique is more efficient than other methods.

KEYWORDS: Integer Partial Differential Equations, Symbolic and Numerical Methods, Shortest Path Problem

1. INTRODUCTION
The problem of finding the shortest path in a graph is a classic problem in computer science. It has many applications, such as in network routing and in the design of integrated circuits. This paper presents a new algorithm for solving this problem. The algorithm is based on a combination of symbolic and numerical methods. It is more efficient than other methods.

2. PRELIMINARIES
Let $G = (V, E)$ be a graph with n vertices and m edges. Let s and t be two vertices in V . The shortest path from s to t is the path with the minimum number of edges. This paper presents a new algorithm for finding this path. The algorithm is based on a combination of symbolic and numerical methods.

3. THE ALGORITHM
The algorithm consists of two main steps. In the first step, the graph is converted into a matrix. In the second step, the matrix is processed to find the shortest path. The algorithm is more efficient than other methods.

4. EXPERIMENTAL RESULTS
The algorithm was tested on a set of graphs. The results show that it is more efficient than other methods. It is able to find the shortest path in a graph with a large number of vertices and edges.

5. CONCLUSION
This paper presents a new algorithm for finding the shortest path in a graph. The algorithm is based on a combination of symbolic and numerical methods. It is more efficient than other methods.

REFERENCES
[1] Dijkstra, E. W. A note on algorithms for solving the shortest path problem. *Mathematische Annalen*, 1959, 159, 371-375.
[2] JHipster, an open-source generator for Web applications. <http://jhipster.github.io/>

CONTACT: Alexandre Nuttack, alexandre.nuttack@unam.ac.be

Information and Software Technology

Scalable prediction of non-functional properties in software product lines: Footprint and memory consumption

Nohbert Siegmund¹, Mario Rosenmüller², Paolo G. Giarrusso³, Sven Apel⁴

¹University of Cologne, Germany
²University of Cologne, Germany
³University of Cologne, Germany
⁴University of Cologne, Germany

ABSTRACT
A software product line is a family of related software products, typically created from a set of common assets. Users select features to build a product that fulfills their needs. Users often expect a product to have specific non-functional properties, such as a small footprint or a bounded response time. Because a product line may have an exponential number of products with respect to the number of features, it is difficult to ensure non-functional properties for each possible product. Therefore, the overall goal is to derive optimal products with respect to non-functional requirements by choosing customers which features will be selected.

KEYWORDS: Feature Configuration, Performance Prediction, Sampling

July 27, 2014. 16:17. WSPC/INSTRUCTION FILE jhipster

Proceeding Paper
© World Scientific Publishing Company

EXPERIMENTS ON OPTIMIZING THE PERFORMANCE OF THE JHIPSTER GENERATOR

ALEXANDRE NUTTACK¹ AND BERTRAND KIEHN² AND CHRISTIAN BARDIOL³ AND NICOLAS BILLOREAU⁴ AND XAVIER DEVREY⁵
¹ IRISA, University of Rennes 1, France
² IRISA, University of Rennes 1, France
³ IRISA, University of Rennes 1, France
⁴ IRISA, University of Rennes 1, France
⁵ IRISA, University of Rennes 1, France

Received April 2014
Revised July 2014
Accepted July 2014
Communicated by Gene Golub

ABSTRACT
A detailed technique for automatically solving integer partial differential equations is presented. This technique is based on a combination of symbolic and numerical methods. It is applied to the problem of finding the shortest path in a graph. The results show that this technique is more efficient than other methods.

KEYWORDS: Integer Partial Differential Equations, Symbolic and Numerical Methods, Shortest Path Problem

1. INTRODUCTION
The problem of finding the shortest path in a graph is a classic problem in computer science. It has many applications, such as in network routing and in the design of integrated circuits. This paper presents a new algorithm for solving this problem. The algorithm is based on a combination of symbolic and numerical methods. It is more efficient than other methods.

2. PRELIMINARIES
Let $G = (V, E)$ be a graph with n vertices and m edges. Let s and t be two vertices in V . The shortest path from s to t is the path with the minimum number of edges. This paper presents a new algorithm for finding this path. The algorithm is based on a combination of symbolic and numerical methods.

3. THE ALGORITHM
The algorithm consists of two main steps. In the first step, the graph is converted into a matrix. In the second step, the matrix is processed to find the shortest path. The algorithm is more efficient than other methods.

4. EXPERIMENTAL RESULTS
The algorithm was tested on a set of graphs. The results show that it is more efficient than other methods. It is able to find the shortest path in a graph with a large number of vertices and edges.

5. CONCLUSION
This paper presents a new algorithm for finding the shortest path in a graph. The algorithm is based on a combination of symbolic and numerical methods. It is more efficient than other methods.

REFERENCES
[1] Dijkstra, E. W. A note on algorithms for solving the shortest path problem. *Mathematische Annalen*, 1959, 159, 371-375.
[2] JHipster, an open-source generator for Web applications. <http://jhipster.github.io/>

CONTACT: Alexandre Nuttack, alexandre.nuttack@unam.ac.be

Empir. Software Eng. (2019) 24:674–717

em all, is it worth it? Assessing configuration on the JHipster Web development stack

lin¹, Alexandre Nuttack², Mathieu Acher³, Devrey⁴, Gilles Perrouin⁵, Benoit Baudry⁶

July 27, 2018
2018

Many approaches for testing configurable software systems start from the same idea: it is impossible to test all configurations. This motivated the definition of y-aware abstractions and sampling techniques to cope with large configuration spaces. There is a theoretical barrier that prevents the exhaustive testing of all configurations: enumerating them is not even semi-decidable. In this paper, we believe there is no way to be learned by systematically and exhaustively testing a configurable system. In this case study, we report on the first ever endeavor to test all configurations of the JHipster web development stack.

Received April 2018
Revised July 2018
Accepted July 2018
Communicated by Gene Golub

ABSTRACT
A software product line is a family of related software products, typically created from a set of common assets. Users select features to build a product that fulfills their needs. Users often expect a product to have specific non-functional properties, such as a small footprint or a bounded response time. Because a product line may have an exponential number of products with respect to the number of features, it is difficult to ensure non-functional properties for each possible product. Therefore, the overall goal is to derive optimal products with respect to non-functional requirements by choosing customers which features will be selected.

KEYWORDS: Feature Configuration, Performance Prediction, Sampling

July 27, 2014. 16:17. WSPC/INSTRUCTION FILE jhipster

Proceeding Paper
© World Scientific Publishing Company

EXPERIMENTS ON OPTIMIZING THE PERFORMANCE OF THE JHIPSTER GENERATOR

ALEXANDRE NUTTACK¹ AND BERTRAND KIEHN² AND CHRISTIAN BARDIOL³ AND NICOLAS BILLOREAU⁴ AND XAVIER DEVREY⁵
¹ IRISA, University of Rennes 1, France
² IRISA, University of Rennes 1, France
³ IRISA, University of Rennes 1, France
⁴ IRISA, University of Rennes 1, France
⁵ IRISA, University of Rennes 1, France

Received April 2014
Revised July 2014
Accepted July 2014
Communicated by Gene Golub

ABSTRACT
A detailed technique for automatically solving integer partial differential equations is presented. This technique is based on a combination of symbolic and numerical methods. It is applied to the problem of finding the shortest path in a graph. The results show that this technique is more efficient than other methods.

KEYWORDS: Integer Partial Differential Equations, Symbolic and Numerical Methods, Shortest Path Problem

1. INTRODUCTION
The problem of finding the shortest path in a graph is a classic problem in computer science. It has many applications, such as in network routing and in the design of integrated circuits. This paper presents a new algorithm for solving this problem. The algorithm is based on a combination of symbolic and numerical methods. It is more efficient than other methods.

2. PRELIMINARIES
Let $G = (V, E)$ be a graph with n vertices and m edges. Let s and t be two vertices in V . The shortest path from s to t is the path with the minimum number of edges. This paper presents a new algorithm for finding this path. The algorithm is based on a combination of symbolic and numerical methods.

3. THE ALGORITHM
The algorithm consists of two main steps. In the first step, the graph is converted into a matrix. In the second step, the matrix is processed to find the shortest path. The algorithm is more efficient than other methods.

4. EXPERIMENTAL RESULTS
The algorithm was tested on a set of graphs. The results show that it is more efficient than other methods. It is able to find the shortest path in a graph with a large number of vertices and edges.

5. CONCLUSION
This paper presents a new algorithm for finding the shortest path in a graph. The algorithm is based on a combination of symbolic and numerical methods. It is more efficient than other methods.

REFERENCES
[1] Dijkstra, E. W. A note on algorithms for solving the shortest path problem. *Mathematische Annalen*, 1959, 159, 371-375.
[2] JHipster, an open-source generator for Web applications. <http://jhipster.github.io/>

CONTACT: Alexandre Nuttack, alexandre.nuttack@unam.ac.be

Efficient and Effective Testing of Automated Software Product Lines

Alexandre Chreyer
Damian A. Bruns, 71063 Siedlitz, Germany

Ray Brühwiler
Coburg University of Applied Sciences, and, Friedrich-Str. 2, 96450 Coburg, Germany

Corresponding author. E-mail: raef.bruehwiler@coburg.de

Received: 18 April 2014; Accepted: 8 May 2014; Published online: 20 May 2014
DOI: 10.1007/s11265-014-9501-0

Abstract: Within the automotive industry, the clients' high demand for individualized customer products results in a number of product variants. In order to control the complexity of developing these variants, a product approach is used that supports reuse of common sets of assets (e.g., requirements, software code, and test cases). This approach is based on the concept of product lines. In this paper, we propose a new test approach based on product lines. The approach is based on the concept of product lines. It is more efficient than other methods.

KEYWORDS: Product Line Testing, Requirements Coverage, Variant Coverage, Automotive Embedded Systems

1. INTRODUCTION
The problem of finding the shortest path in a graph is a classic problem in computer science. It has many applications, such as in network routing and in the design of integrated circuits. This paper presents a new algorithm for solving this problem. The algorithm is based on a combination of symbolic and numerical methods. It is more efficient than other methods.

2. PRELIMINARIES
Let $G = (V, E)$ be a graph with n vertices and m edges. Let s and t be two vertices in V . The shortest path from s to t is the path with the minimum number of edges. This paper presents a new algorithm for finding this path. The algorithm is based on a combination of symbolic and numerical methods.

3. THE ALGORITHM
The algorithm consists of two main steps. In the first step, the graph is converted into a matrix. In the second step, the matrix is processed to find the shortest path. The algorithm is more efficient than other methods.

4. EXPERIMENTAL RESULTS
The algorithm was tested on a set of graphs. The results show that it is more efficient than other methods. It is able to find the shortest path in a graph with a large number of vertices and edges.

5. CONCLUSION
This paper presents a new algorithm for finding the shortest path in a graph. The algorithm is based on a combination of symbolic and numerical methods. It is more efficient than other methods.

REFERENCES
[1] Dijkstra, E. W. A note on algorithms for solving the shortest path problem. *Mathematische Annalen*, 1959, 159, 371-375.
[2] JHipster, an open-source generator for Web applications. <http://jhipster.github.io/>

CONTACT: Alexandre Nuttack, alexandre.nuttack@unam.ac.be

A multi-objective test data generation approach for mutation testing of feature models

Rui A. Mamede Filho¹ and Silvio R. Vergilio

University of São Paulo, Brazil
Email: rafilho@usp.br, svergilio@usp.br

Received: 18 April 2014; Accepted: 8 May 2014; Published online: 20 May 2014
DOI: 10.1007/s11265-014-9501-0

Abstract: Mutation approaches have been recently applied for feature testing of Software Product Lines (SPLs). The idea is to select test cases, associated to mutation operators that describe problem faults in the Feature Model (FM). In this work, the operators and mutation score can be used to evaluate and generate a test set that is a set of SPL products to be tested. However, the generation of test sets to kill all the mutants with a reduced, possible minimum, number of products is a complex task.

KEYWORDS: Mutation Testing, Multi-objective Optimization, Software Product Line

1. Background
A Software Product Line (SPL) can be defined as a set of products that share common features (Fonseca et al. 2005). A feature is defined as an active functionality or system attributes that are visible to the user. Features add distinguishing products and are important to represent variability. In some domains, products can be generated by selecting different features. The features are generally expressed in a Feature Model (FM), which allows a hierarchical arrangement of features represented by a tree.

The adoption of the SPL approach in industries is covered (SPL 2004). Due to the associated advantages. With this increasing usage, the demand for SPL testing techniques has also been growing. In this context, feature models have become a key element in the feature testing, which tests the products derived from the FM.

The SPL approach is based on the concept of product lines. It is more efficient than other methods.

2. Preliminaries
Let $G = (V, E)$ be a graph with n vertices and m edges. Let s and t be two vertices in V . The shortest path from s to t is the path with the minimum number of edges. This paper presents a new algorithm for finding this path. The algorithm is based on a combination of symbolic and numerical methods.

3. The Algorithm
The algorithm consists of two main steps. In the first step, the graph is converted into a matrix. In the second step, the matrix is processed to find the shortest path. The algorithm is more efficient than other methods.

4. Experimental Results
The algorithm was tested on a set of graphs. The results show that it is more efficient than other methods. It is able to find the shortest path in a graph with a large number of vertices and edges.

5. Conclusion
This paper presents a new algorithm for finding the shortest path in a graph. The algorithm is based on a combination of symbolic and numerical methods. It is more efficient than other methods.

References
[1] Dijkstra, E. W. A note on algorithms for solving the shortest path problem. *Mathematische Annalen*, 1959, 159, 371-375.
[2] JHipster, an open-source generator for Web applications. <http://jhipster.github.io/>

Contact: Alexandre Nuttack, alexandre.nuttack@unam.ac.be

Journal 23(October 2018) | Special Issue 2018 | November 27, 2018
 SPECIAL ISSUE PAPER
 Performance-influence models of multigrid methods: A case study on triangular grids
 Alexander Grebbahn¹ | Carries Rodrigo² | Norbert Siegmund³ | Francisco J. Gaspar⁴ | Sven Apel⁵

Abstract
 Multigrid methods are among the most efficient algorithms for solving elliptical partial differential equations. Typically a multigrid solver either refines configuration space or time performance for different configurations and feature points. However, having a good understanding of the performance influence of multigrid parameters is essential for the performance of the solver. In this paper, we propose a classification for product sampling techniques and classify the existing literature accordingly. We distinguish the important characteristics of such approaches based on the information used for sampling, the kind of algorithm, and the achieved coverage criteria. Furthermore, we give an overview on existing tools and evaluations of product sampling techniques. We share our insights on the state-of-the-art of product sampling and discuss potential future work.

Keywords
 Software product lines, product sampling, feature models, configuration space, performance analysis, testing, SaaS, Cloud, DevOps

1 INTRODUCTION
 Many important applications, ranging from scientific simulations and industrial applications to games in computers and mobile devices, are mathematically modeled to solve differential equations (PDE). The complexity of the state-of-the-art algorithms, solutions or iterative algorithms, approximates the underlying continuous problems, giving rise to large sparse systems of algebraic equations, which need to be solved numerically. Since their introduction in the 1970s, multigrid methods have become a standard tool for solving elliptical partial differential equations. The main reason for this is that multigrid methods are able to solve elliptical partial differential equations efficiently. The main reason for this is that multigrid methods are able to solve elliptical partial differential equations efficiently. The main reason for this is that multigrid methods are able to solve elliptical partial differential equations efficiently.

Yo Variability!
JHipster: A Playground for Web-Apps

Axel Hellin¹, Alexandre Nuttinck², Mathieu Acher³, Xavier Devroey⁴, Gilles Perrouin⁵, Benoit Baudry⁶

A Classification of Product Sampling for Software Product Lines

Mahsa Varshosaz,¹ Mustafa Al-Hajjaji,² Thomas Thüm,³ Tobias Runge,³ Mohammad Reza Mousavi,^{4,1} and Ina Schaefer³
¹Halmstad University, Sweden ²Pure-Systems GmbH, Germany ³TU Braunschweig, Germany ⁴University of Leicester, UK

ABSTRACT
 The analysis of software product lines is challenging due to the potentially large number of products, which grow exponentially in terms of the number of features. Product sampling is a technique used to avoid exhaustive testing, which is often infeasible. In this paper, we propose a classification for product sampling techniques and classify the existing literature accordingly. We distinguish the important characteristics of such approaches based on the information used for sampling, the kind of algorithm, and the achieved coverage criteria. Furthermore, we give an overview on existing tools and evaluations of product sampling techniques. We share our insights on the state-of-the-art of product sampling and discuss potential future work.

CCS CONCEPTS
 • Software and its engineering → Software product lines.

KEYWORDS
 Sampling Algorithms, Software Product Lines, Testing, Feature Interaction, Domain Models

ACM Reference Format:
 Mahsa Varshosaz,¹ Mustafa Al-Hajjaji,² Thomas Thüm,³ Tobias Runge,³ Mohammad Reza Mousavi,^{4,1} and Ina Schaefer³. 2018. A Classification of Product Sampling for Software Product Lines. In *SPLC '18: 22nd International Systems and Software Product Line Conference, September 10–14, 2018, Gothenburg, Sweden*. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3233027.3233035>

1 INTRODUCTION
 Software product lines (SPLs) have become common practice for mass production and customization of software systems. In an SPL, products are developed based on a common core. The main goal of using SPLs is to enable systematic reuse in different phases of development by considering the commonalities and variabilities among the products in an SPL [50].

Testing and analysis of software product lines is known to be challenging due to the sheer number of possible products, which makes exhaustive testing and analysis practically impossible. To

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. 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 or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
 SPLC '18, September 10–14, 2018, Gothenburg, Sweden
 © 2018 ACM Association for Computing Machinery.

avoid this problem, one may resort to product sampling techniques [82] that provide a subset of all valid products. These products are supposed to collectively cover the behavior of the product line and hence for example testing them should reveal most faults in all other products.

Several approaches have been proposed for product sampling in the context of software product lines, in order to search the vast space of valid products [28, 56, 66]. For such approaches, a myriad of search algorithms for finding a sample to cover a product line have been proposed, where the notion of coverage may also vary from one approach to another. Different algorithms use different types of information sources to find a covering sample. Moreover, the proposed algorithms have typically been evaluated with respect to different criteria and with different degrees of tool support and reproducibility.

We aim for bringing more structure onto the extensive literature on product sampling. In contrast to existing surveys on product sampling [49, 71] or product-line testing [56], we do not follow a systematic process in which all interesting research questions is defined up-front. In contrast, our goal is to provide more insights for readers by means of a detailed classification of existing sampling techniques. We envision that our insights can be used to have a better understanding of such techniques for education and research and also for recognizing their requirements and shortcomings to apply such techniques in practice. To this end, we considered a literature catalog with 48 publications [1–48]. We limited our search to find studies that are focusing on new sampling algorithms [1–38] or evaluations of existing ones [39–48].¹

Our contributions are threefold. First, we propose a classification for product sampling, involving input data used for sampling, the actual algorithm and achieved coverage, as well as its evaluation (cf. Section 3). Second, we survey and classified the literature with respect to the classification (cf. Section 4–6). The list of studies and their classification can be found online.² We plan to update this list in the future and welcome any pointers by the community. Third, we identify underrepresented research areas to be addressed by future work.

Our synthesis results indicate that most techniques used problem-space input information, in terms of feature models in generating product samples. Solution space information, such as test artifacts or code coverage, has rarely been used and we think bridging this gap may lead to novel research results. Regarding the developed techniques and algorithms, greedy and evolutionary algorithms have been developed most in this domain. Also, there are no techniques that consider the history of feature models and evolution in software product lines. Regarding evaluation, there are very few

¹References are sorted by author names but grouped into proposed algorithms, evalua-



Test them all, is it worth it? Assessing configuration sampling on the JHipster Web development stack

Axel Hellin¹, Alexandre Nuttinck², Mathieu Acher³, Xavier Devroey⁴, Gilles Perrouin⁵, Benoit Baudry⁶

Published online: 17 July 2018
 © The Author(s) 2018

Abstract Many approaches for testing configurable software systems start from the same assumption: it is impossible to test all configurations. This motivated the definition of variability-aware abstractions and sampling techniques to cope with large configuration spaces. Yet, there is no theoretical barrier that prevents the exhaustive testing of all configurations by simply enumerating them if the effort required to do so remains acceptable. Not this, we believe there is a lot to be learned by systematically and exhaustively testing a configurable system. In this case study, we report on the first ever endeavor to test

Keywords: Configuration sampling, variability-aware abstractions, testing, SaaS, Cloud, DevOps

1 INTRODUCTION
 Many important applications, ranging from scientific simulations and industrial applications to games in computers and mobile devices, are mathematically modeled to solve differential equations (PDE). The complexity of the state-of-the-art algorithms, solutions or iterative algorithms, approximates the underlying continuous problems, giving rise to large sparse systems of algebraic equations, which need to be solved numerically. Since their introduction in the 1970s, multigrid methods have become a standard tool for solving elliptical partial differential equations. The main reason for this is that multigrid methods are able to solve elliptical partial differential equations efficiently. The main reason for this is that multigrid methods are able to solve elliptical partial differential equations efficiently.

Product Sampling for Product Lines: The Scalability Challenge

Thomas Thüm¹, Tobias Runge², Ina Schaefer³

1 INTRODUCTION
 Software product lines (SPLs) have become common practice for mass production and customization of software systems. In an SPL, products are developed based on a common core. The main goal of using SPLs is to enable systematic reuse in different phases of development by considering the commonalities and variabilities among the products in an SPL [50].

“A Classification of Product Sampling for Software Product Lines” Varshosaz, Al-Hajjaji, Thüm, Runge, Mousavi, Schaefer SPLC 2018



A Classification of Product Sampling for Software Product Lines

Mahsa Varshosaz,¹ Mustafa Al-Hajjaji,² Thomas Thüm,³ Tobias Runge,³
Mohammad Reza Mousavi,^{4,1} and Ina Schaefer¹
¹Halmstad University, Sweden ²Pure-Systems GmbH, Germany
³TU Braunschweig, Germany ⁴University of Leicester, UK

ABSTRACT

The analysis of software product lines is challenging due to the potentially large number of products, which grow exponentially in terms of the number of features. Product sampling is a technique used to avoid exhaustive testing, which is often infeasible. In this paper, we propose a classification for product sampling techniques and classify the existing literature accordingly. We distinguish the important characteristics of such approaches based on the information used for sampling, the kind of algorithm, and the achieved coverage criteria. Furthermore, we give an overview on existing tools and evaluations of product sampling techniques. We share our insights on the state-of-the-art of product sampling and discuss potential future work.

CCS CONCEPTS

Software and its engineering → Software product lines:

KEYWORDS

Sampling Algorithms, Software Product Lines, Testing, Feature Interaction, Domain Models

ACM Reference Format:

Mahsa Varshosaz,¹ Mustafa Al-Hajjaji,² Thomas Thüm,³ Tobias Runge,³ Mohammad Reza Mousavi,^{4,1} and Ina Schaefer¹. 2018. A Classification of Product Sampling for Software Product Lines. In *SPLC '18: 22nd International Systems and Software Product Line Conference*, September 10–14, 2018, Gothenburg, Sweden. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3233027.3233035>

1 INTRODUCTION

Software product lines (SPLs) have become common practice for mass production and customization of software systems. In an SPL, products are developed based on a common core. The main goal of using SPLs is to enable systematic reuse in different phases of development by considering the commonalities and variabilities among the products in an SPL [50].

Testing and analysis of software product lines is known to be challenging due to the sheer number of possible products, which makes exhaustive testing and analysis practically impossible. To

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. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org.
SPLC '18, September 10–14, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6664-5/18/09...\$11.00
<https://doi.org/10.1145/3233027.3233035>

alleviate this problem, one may resort to product sampling techniques [82] that provide a subset of all valid products. These products are supposed to collectively cover the behavior of the product line and hence for example testing them should reveal most faults in all other products.

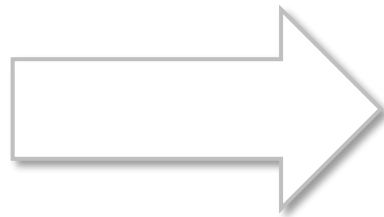
Several approaches have been proposed for product sampling in the context of software product lines, in order to search the vast space of valid products [28, 56, 66]. For such approaches, a myriad of search algorithms for finding a sample to cover a product line have been proposed, where the notion of coverage may also vary from one approach to another. Different algorithms use different types of information sources to find a covering sample. Moreover, the proposed algorithms have typically been evaluated with respect to different criteria and with different degrees of tool support and reproducibility.

We aim for bringing more structure onto the extensive literature on product sampling. In contrast to existing surveys on product sampling [49, 71] or product-line testing [56], we do not follow a systematic process in which all interesting research questions is defined up-front. In contrast, our goal is to provide more insights for readers by means of a detailed classification of existing sampling techniques. We envision that our insights can be used to have a better understanding of such techniques for education and research and also for recognizing their requirements and shortcomings to apply such techniques in practice. To this end, we considered a literature catalog with 48 publications [1–48]. We limited our search to find studies that are focusing on new sampling algorithms [1–38] or evaluations of existing ones [39–48].¹

Our contributions are threefold. First, we propose a classification for product sampling, involving input data used for sampling, the actual algorithm and achieved coverage, as well as its evaluation (cf. Section 3). Second, we surveyed and classified the literature with respect to the classification (cf. Section 4–6). The list of studies and their classification can be found online.² We plan to update this list in the future and welcome any pointers by the community. Third, we identify underrepresented research areas to be addressed by future work.

Our synthesis results indicate that most techniques used problem-space input information, in terms of feature models in generating product samples. Solution space information, such as test artifacts or code coverage, has rarely been used and we think bridging this gap may lead to novel research results. Regarding the developed techniques and algorithms, greedy and evolutionary algorithms have been developed most in this domain. Also, there are no techniques that consider the history of feature models and evolution in software product lines. Regarding evaluation, there are very few

¹References are sorted by author names but grouped into proposed algorithms, evaluations of sampling algorithms, and other references.
²<http://thomas.thuem-thu.de/sampling/>



Institute of Software Engineering and Automotive Informatics Product Sampling for Software Product Lines



The analysis of software product lines is challenging due to the potentially large number of products, which grow exponentially in terms of the number of features. Product sampling is a technique used to avoid exhaustive testing, which is often infeasible. We have proposed a classification for product sampling techniques and classified the existing literature accordingly. We distinguish the important characteristics of such approaches based on the information used for sampling, the kind of algorithm, and the achieved coverage criteria. Furthermore, we give an overview on existing tools and evaluations of product sampling techniques. We share our insights on the state-of-the-art of product sampling and discuss potential future work.

This website is accepted as an official ACM artifact to our publication at SPLC'18 [1]. As the large number of algorithms and our detailed classification give rise to a large space, that is hard to explore only by means of a table in the proceedings. Here, we offer an interactive table with filter and sorting capabilities. Furthermore, product sampling is a very active research area and new algorithms are published several times a year. We aim to keep the table up-to-date, but kindly ask you to [report any missing literature and wrong classifications to us](#). Instead of sending an e-mail it is also possible to propose changes by means of a [pull request](#). In the request, you can add the paper to the file of Bibtex entries, which is used to generate this website. Please note that there is a dedicated entry called [sampling-tags](#) containing the classification (e.g. feature_model as input data or name-NewAlgorithm to specify the entry in column entitled Algorithm). We will review the changes manually and update the website accordingly.

[1] Mahsa Varshosaz, Mustafa Al-Hajjaji, Thomas Thüm, Tobias Runge, Mohammadreza Mousavi, and Ina Schaefer. [A Classification of Product Sampling for Software Product Lines](#). In *Proceedings of the International Software Product Line Conference (SPLC)*, September 2018. To appear.

Active Filters: No filters applied

Show All entries Search:

Authors	Venue	Year	Title	Algorithm	Input Data	Algorithm Category	Coverage	Evaluation	Application	Further Tags
All	All	All	All	All	All	All	All	All	All	All
Abdel Salam Sayyed, Joseph Ingram, Tim Menzies, Hany Ammar	ASE	2013	Scalable Product Line Configuration: A Straw to Break the Camel's Back							usage of JMetal/Z3
Alireza Ensán, Ebrahim Bagheri, Mohsen Asadi, Dragan Gasevic, Yevgen Biletskiy	ITNG	2011	Goal-Oriented Test Case Selection and Prioritization for Product Line Feature Models	Ensan's algorithm	feature model, expert knowledge	greedy, semi-automatic selection	no coverage guarantee	testing efficiency, evaluation	testing	SPLC18
Anastasia Cmyrev, Ralf Reissing	IJAST	2014	Efficient and Effective Testing of Automotive Software Product Lines	Cmyrev's greedy algorithm	feature model	greedy, semi-automatic selection	feature-wise coverage, requirements coverage	sampling efficiency, testing efficiency, effectiveness, unavailable tool, evaluation	testing	compared to MoSo-PoLiTe, SPLC18
Anastasia Cmyrev, Ralf Reissing	IJAST	2014	Efficient and Effective Testing of Automotive	Cmyrev's simulated annealing	feature model	local search, semi-automatic selection	no coverage guarantee	sampling efficiency, testing efficiency,	testing	compared to MoSo-PoLiTe, SPLC18

<https://tubs-isf.github.io/>

Institute of Software Engineering and Automotive Informatics

Product Sampling for Software Product Lines



The analysis of software product lines is challenging due to the potentially large number of products, which grow exponentially in terms of the number of features. Product sampling is a technique used to avoid exhaustive testing, which is often infeasible. We have proposed a classification for product sampling techniques and classified the existing literature accordingly. We distinguish the important characteristics of such approaches based on the information used for sampling, the kind of algorithm, and the achieved coverage criteria. Furthermore, we give an overview on existing tools and evaluations of product sampling techniques. We share our insights on the state-of-the-art of product sampling and discuss potential future work.

This website is accepted as an official ACM artifact to our publication at SPLC'18 [1]. As the large number of algorithms and our detailed classification give rise to a large space, that is hard to explore only by means of a table in the proceedings. Here, we offer an interactive table with filter and sorting capabilities. Furthermore, product sampling is a very active research area and new algorithms are published several times a year. We aim to keep the table up-to-date, but kindly ask you to [report any missing literature and wrong classifications to us](#). Instead of sending an e-mail it is also possible to propose changes by means of a [pull request](#). In the request, you can add the paper to the file of Bibtex entries, which is used to generate this website. Please note that there is a dedicated entry called `sampling-tags` containing the classification (e.g., `feature model` as input data or `name=NewAlgorithm` to specify the entry in column entitled Algorithm). We will review the changes manually and update the website accordingly.

[1] Mahsa Varshosaz, Mustafa Al-Hajjaji, Thomas Thüm, Tobias Runge, Mohammadreza Mousavi, and Ina Schaefer. [A Classification of Product Sampling for Software Product Lines](#). In *Proceedings of the International Software Product Line Conference (SPLC)*. September 2018. To appear.

Active Filters: No filters applied

Show All entries

Search:

Authors	Venue	Year	Title	Algorithm	Input Data	Algorithm Category	Coverage	Evaluation	Application	Further Tags
All	All	All		All	All	All	All	All	All	All
Abdel Salam Sayyad, Joseph Ingram, Tim Menzies, Hany Ammar	ASE	2013	Scalable Product Line Configuration: A Straw to Break the Camel's Back							usage of jMetal/Z3
Alireza Ensan, Ebrahim Bagheri, Mohsen Asadi, Dragan Gasevic, Yevgen Biletskiy	ITNG	2011	Goal-Oriented Test Case Selection and Prioritization for Product Line Feature Models	Ensan's algorithm	feature model, expert knowledge	greedy, semi-automatic selection	no coverage guarantee	testing efficiency, evaluation	testing	SPLC18
Anastasia Smyrev, Ralf Reissing	IJAST	2014	Efficient and Effective Testing of Automotive Software Product Lines	Smyrev's greedy algorithm	feature model	greedy, semi-automatic selection	feature-wise coverage, requirements coverage	sampling efficiency, testing efficiency, effectiveness, unavailable tool, evaluation	testing	compared to MoSo-PoLiTe, SPLC18
Anastasia Smyrev, Ralf Reissing	IJAST	2014	Efficient and Effective Testing of Automotive	Smyrev's simulated annealing	feature model	local search, semi-automatic selection	no coverage guarantee	sampling efficiency, testing efficiency,	testing	compared to MoSo-PoLiTe, SPLC18

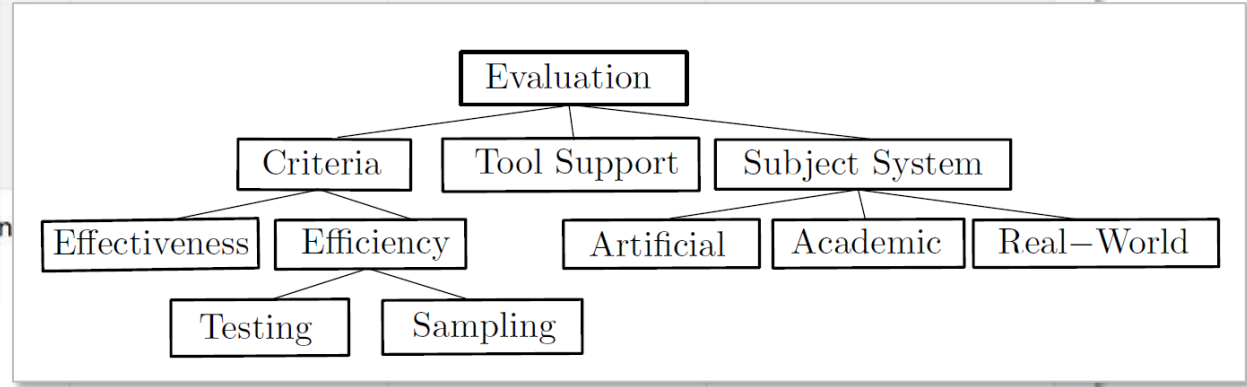
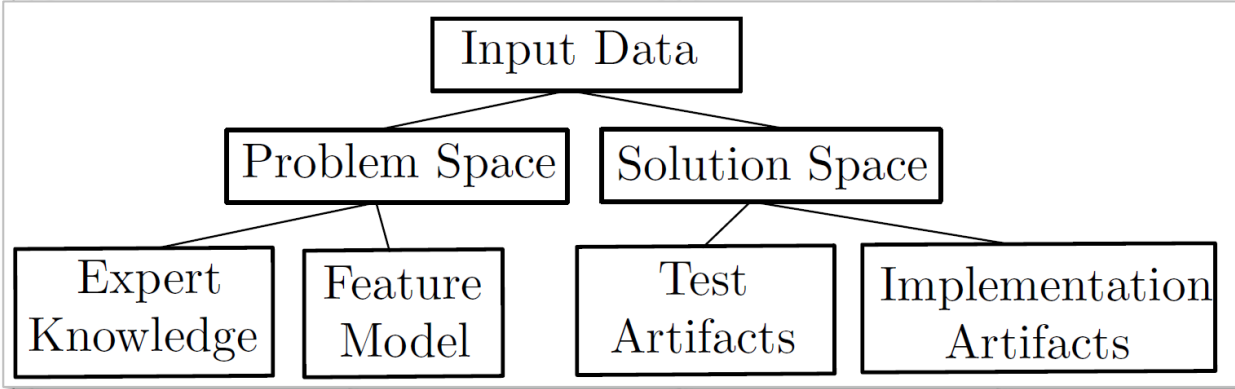
<https://tubs-isf.github.io/>

Algorithm	Input Data	Algorithm Category	Coverage	Evaluation	Application	Further Tags
All	All	All	All	All	All	All
						usage of jMetal/Z3
Ensan's algorithm	feature model, expert knowledge	greedy, semi-automatic selection	no coverage guarantee	testing efficiency, evaluation	testing	SPLC18
Cmyrev's greedy algorithm	feature model	greedy, semi-automatic selection	feature-wise coverage, requirements coverage	sampling efficiency, testing efficiency, effectiveness, unavailable tool, evaluation	testing	compared to MoSo-PoLiTe, SPLC18
Cmyrev's simulated annealing	feature model	local search, semi-automatic selection	no coverage guarantee	sampling efficiency, testing efficiency,	testing	compared to MoSo-PoLiTe, SPLC18

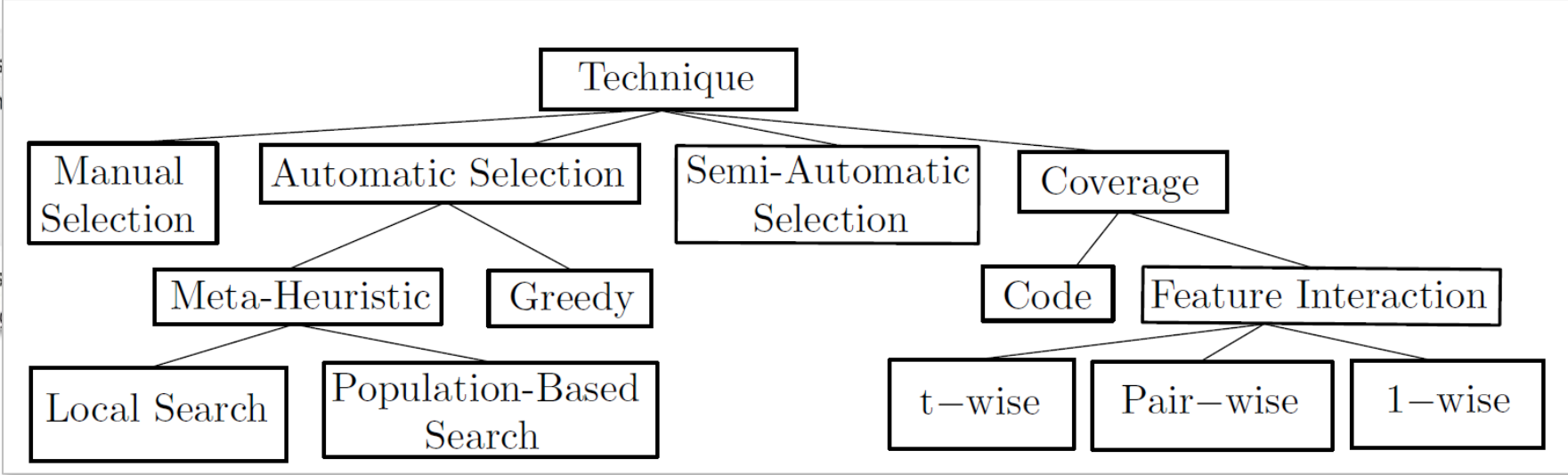
<https://tubs-isf.github.io/>

Algorithm	Input Data	Algorithm Category	Coverage	Evaluation	Application	Further Tags
-----------	------------	--------------------	----------	------------	-------------	--------------

All All All All



Cmyrev's algorithm
Cmyrev's annealing



compared to MoSo-PoLiTe, SPLC18
compared to MoSo-PoLiTe, SPLC18

<https://tubs-isf.github.io/>



A Classification of Product Sampling for Software Product Lines

Mahsa Varshosaz,¹ Mustafa Al-Hajjaji,² Thomas Thüm,³ Tobias Runn

Mohammad Reza Mousavi,^{4,1} and Ina Schaefer^{2,3}

¹Halmstad University, Sweden ²Pure-Systems GmbH, Germany

³TU Braunschweig, Germany ⁴University of Leicester, UK

ABSTRACT

The analysis of software product lines is challenging due to the potentially large number of products, which grow exponentially in terms of the number of features. Product sampling is a technique used to avoid exhaustive testing, which is often infeasible. In this paper, we propose a classification for product sampling techniques and classify the existing literature accordingly. We distinguish the important characteristics of such approaches based on the information used for sampling, the kind of algorithm, and the achieved coverage criteria. Furthermore, we give an overview on existing tools and evaluations of product sampling techniques. We share our insights on the state-of-the-art of product sampling and discuss potential future work.

CCS CONCEPTS

• Software and its engineering → Software product lines

KEYWORDS

Sampling Algorithms, Software Product Lines, Testing, Feature Interaction, Domain Models

ACM Reference Format:

Mahsa Varshosaz,¹ Mustafa Al-Hajjaji,² Thomas Thüm,³ Tobias Runn, Mohammad Reza Mousavi,^{4,1} and Ina Schaefer^{2,3}. 2020. A Classification of Product Sampling for Software Product Lines. In *SPLC ’20 (2nd International Systems and Software Product Line Conference, September 14, 2018, Gothenburg, Sweden, ACM, New York, NY, USA), 13 pages*. <https://doi.org/10.1145/3233027.3233035>

1 INTRODUCTION

Software product lines (SPL) have become a common practice for mass production and customization of software systems. In an SPL, products are developed based on a common core. The main goal of using SPLs is to ease system development in different phases of development by considering commonalities and variabilities among the products in an SPL.

Testing and analysis of software product lines is known to be challenging due to the sheer number of possible products, which makes exhaustive testing and analysis practically impossible. To

alleviate this problem, one must resort to product sampling techniques [82] that provide a small set of valid products. These products are supposed to effectively represent the behavior of the product line and hence for a simple testing effort should reveal most faults in all other products.

Several approaches have been proposed for product sampling in the context of software product lines in order to search the vast space of valid products [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81]. For such approaches, a myriad of search algorithms for finding a sample to cover a product line have been proposed, where the notion of coverage may also vary from one approach to another. Different algorithms use different sources of information to find a covering sample. Moreover, these approaches have typically been evaluated with respect to their effectiveness and with different degrees of tool support and

We aim at bringing more structure onto the extensive literature on product sampling. In contrast to existing surveys on product sampling [49, 71] or product-line testing [56], we do not follow a systematic process in which all interesting research questions are listed up-front. In contrast, our goal is to provide more insights to the readers by means of a detailed classification of existing sampling techniques. We envision that our insights can be used to have a better understanding of such techniques for education and research and also for recognizing their requirements and shortcomings to apply such techniques in practice. To this end, we considered a literature catalog with 48 publications [1–48]. We limited our search to find studies that are focusing on new sampling algorithms [1–38] or evaluations of existing ones [39–48].¹

Our contributions are threefold. First, we propose a classification for product sampling, involving input data used for sampling, the actual algorithm and achieved coverage, as well as its evaluation (cf. Section 3). Second, we surveyed and classified the literature with respect to the classification (cf. Section 4–6). The list of studies and their classification can be found online.² We plan to update this list in the future and welcome any pointers by the community. Third, we identify underrepresented research areas to be addressed by future work.

Our synthesis results indicate that most techniques used problem-space input information, in terms of feature models in generating product samples. Solution space information, such as test artifacts or code coverage, has rarely been used and we think bridging this gap may lead to novel research results. Regarding the developed techniques and algorithms, greedy and evolutionary algorithms have been developed most in this domain. Also, there are no techniques that consider the history of feature models and evolution in software product lines. Regarding evaluation, there are very few

¹References are sorted by author names but grouped into proposed algorithms, evaluations of sampling algorithms, and other references.
²<http://thomas.tn-ilm-isa.de/sampling>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted by ACM, provided that the copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation information on the first page. Copyrights for components of this work owned by others than ACM must be honored. Copying beyond the limits permitted by Sections 107 and 108 of the 1976 Copyright Act requires prior permission of the copyright owner. For all other uses, permission should be sought from ACM. Copyright © 2020, ACM. ISBN 978-1-4503-7323-3/20/09...\$15.00 <https://doi.org/10.1145/3233027.3233035>

What are the latest trends?



A Classification of Product Sampling for Software Product Lines

Mahsa Varshosaz,¹ Mustafa Al-Hajjaji,² Tobias Runge,³ Tobias Hooge,⁴ Mohammad Reza Mousavi,^{4,5} and Ina Schaefer^{2,6}

ABSTRACT
The analysis of software product lines (SPLs) is a challenging task due to the potentially large number of variants. Exhaustive testing and analysis is often infeasible. In this paper, we propose a classification of product sampling techniques and classify the existing techniques based on the information characterizing the search space, the achieved coverage criteria, and the evaluation tools and evaluation techniques. We share our insights on the state-of-the-art of product sampling and discuss potential future work.

CCS CONCEPTS
• Software and its engineering → Software product lines

KEYWORDS
Sampling Algorithms, Software Product Lines, Testing, Feature Interaction, Domain Models

ACM Reference Format:
Mahsa Varshosaz,¹ Mustafa Al-Hajjaji,² Tobias Runge,³ Tobias Hooge,⁴ Mohammad Reza Mousavi,^{4,5} and Ina Schaefer^{2,6}. 2023. A Classification of Product Sampling for Software Product Lines. In *SPLC '23: Proceedings of the ACM Conference on Software Product Lines*. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3233027.3233035>

1 INTRODUCTION
Software product lines (SPLs) have become common products developed based on a common core. The main goal of using SPLs is to enable systematic reuse in different phases of development by considering the commonalities and variabilities among the products in an SPL [54]. Testing and analysis of software product lines is known to be challenging due to the sheer number of possible products, which makes exhaustive testing and analysis practically impossible. To make digital or hard copies of all or part of this work for personal or commercial use, permission is granted by ACM to individuals to photocopy, reproduce, and distribute reprints for noncommercial use, provided that the fee of \$12.00 is paid directly to ACM. For those organizations that have been granted a photocopy licence by ACM, a separate system of payment has been arranged. The fee code for users of the Transactional Reporting Service is 0730-0816/2023/03-0000-00. 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 or to redistribute to lists, requires prior specific permission from ACM. <https://doi.org/10.1145/3233027.3233035>

© 2023 Association for Computing Machinery.
ACM ISBN 978-1-59593-646-3/18/09...\$15.00
<https://doi.org/10.1145/3233027.3233035>

Did the application areas change?

?

Is this classification sufficient?

Golden Set

A Classification of Product Sampling for Software Product Lines

Mahsa Varshosaz,¹ Mustafa Al-Hajjaji,² Thomas Thüm,³ Tobias Runge,³
Mohammad Reza Mousavi,^{4,1} and Ina Schaefer³
¹Halmstad University, Sweden ²Pure-Systems GmbH, Germany
³TU Braunschweig, Germany ⁴University of Leicester, UK

ABSTRACT

The analysis of software product lines is challenging due to the potentially large number of products, which grow exponentially in terms of the number of features. Product sampling is a technique used to avoid exhaustive testing, which is often infeasible. In this paper, we propose a classification for product sampling techniques and classify the existing literature accordingly. We distinguish the important characteristics of such approaches based on the information used for sampling, the kind of algorithm, and the achieved coverage criteria. Furthermore, we give an overview on existing tools and evaluations of product sampling techniques. We share our insights on the state-of-the-art of product sampling and discuss potential future work.

CCS CONCEPTS

• Software and its engineering → Software product lines.

KEYWORDS

Sampling Algorithms, Software Product Lines, Testing, Feature Interaction, Domain Models

ACM Reference Format:

Mahsa Varshosaz,¹ Mustafa Al-Hajjaji,² Thomas Thüm,³ Tobias Runge,³ Mohammad Reza Mousavi,^{4,1} and Ina Schaefer³. 2018. A Classification of Product Sampling for Software Product Lines. In *SPLC '18: 22nd International Systems and Software Product Line Conference, September 10-14, 2018, Gothenburg, Sweden*. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/325827.325835>

1 INTRODUCTION

Software product lines (SPLs) have become common practice for mass production and customization of software systems. In an SPL, products are developed based on a common core. The main goal of custom SPLs is to enable systematic reuse in different phases of

alleviate this problem, one may resort to product sampling techniques [82] that provide a subset of all valid products. These products are supposed to collectively cover the behavior of the product line and hence for example testing them should reveal most faults in all other products.

Several approaches have been proposed for product sampling in the context of software product lines, in order to search the vast space of valid products [28, 56, 66]. For such approaches, a myriad of search algorithms for finding a sample to cover a product line have been proposed, where the notion of coverage may also vary from one approach to another. Different algorithms use different types of information sources to find a covering sample. Moreover, the proposed algorithms have typically been evaluated with respect to different criteria and with different degrees of tool support and reproducibility.

We aim for bringing more structure onto the extensive literature on product sampling. In contrast to existing surveys on product sampling [49, 71] or product-line testing [56], we do not follow a systematic process in which all interesting research questions is defined up-front. In contrast, our goal is to provide more insights for readers by means of a detailed classification of existing sampling techniques. We envision that our insights can be used to have a better understanding of such techniques for education and research and also for recognizing their requirements and shortcomings to apply such techniques in practice. To this end, we considered a literature catalog with 49 publications [1–48]. We limited our search to find studies that are focusing on new sampling algorithms [1–38] or evaluations of existing ones [39–48].¹

Our contributions are threefold. First, we propose a classification for product sampling, involving input data used for sampling, the actual algorithm and achieved coverage, as well as its evaluation (cf. Section 3). Second, we surveyed and classified the literature with respect to the classification (cf. Section 4–6). The list of studies and their classification can be found online.² We plan to update this list as the features and evaluation are mature, by the community.³ Third,



Product Sampling for Product Lines: The Scalability Challenge

Tobias Pett
tpett@tu-braunschweig.de
TU Braunschweig
Germany

Thomas Thüm
tthuem@tu-braunschweig.de
TU Braunschweig
Germany

Tobias Runge
tobias.runge@tu-braunschweig.de
TU Braunschweig
Germany

Sebastian Krieter
sebastian.krieter@ovgu.de
University of Magdeburg
Germany

Malte Lochau
malte.lochau@es.tu-darmstadt.de
TU Darmstadt
Germany

Ina Schaefer
ischaef@tu-braunschweig.de
TU Braunschweig
Germany

ABSTRACT

Quality assurance for product lines is often infeasible for each product separately. Instead, only a subset of all products (i.e., a sample) is considered during testing such that at least the coverage of certain feature interactions is guaranteed. While pair-wise interaction sampling only covers all interactions between two features, its generalization to t -wise interaction sampling ensures coverage for all interactions among t features. However, sampling large product lines poses a challenge, as today's algorithms tend to run out of memory, do not terminate, or produce samples, which are too large to be tested. To initiate a community effort, we provide a set of large real-world feature models with up to 19 thousand features, which are supposed to be sampled. The performance of sampling approaches is evaluated based on the CPU time and memory consumed to retrieve a sample, the sample size for a given coverage (i.e. the value of t) and whether the sample achieves full t -wise coverage. A well-performing sampling algorithm achieves full t -wise coverage, while minimizing the other properties as best as possible.

CCS CONCEPTS

• Software and its engineering → Software product lines.

KEYWORDS

software product lines, product line testing, product sampling, real-world feature models

ACM Reference Format:

Tobias Pett, Thomas Thüm, Tobias Runge, Sebastian Krieter, Malte Lochau,

1 INTRODUCTION

Modern software engineering struggles with increasing requirements regarding finer customization and faster time to market. A common solution is to design systems as software product lines. This way a collection of customized products can be built based on a common core [5]. However, the variability contained in a software product line poses a potentially large number of possible products. Analyzing all products individually is infeasible in most cases [30]. Therefore, quality assurance is often performed on a small subset of product configurations, which presumably covers a sufficient amount of functionality of the product line.

A common technique to build a subset of products is *combinatorial interaction sampling (CIT)* [17]. The goal of combinatorial interaction testing is to build samples, which achieve t -wise interaction coverage between sets of features. T -wise interaction coverage requires that every possible combination of t features is covered by at least one product in the sample. Commonly used coverage criteria include feature-wise coverage ($t = 1$), pair-wise coverage ($t = 2$), or three-wise coverage ($t = 3$).

Over the years, many sampling algorithms have been developed to generate samples that achieve feature interaction coverage. To generate a sample different input artifacts such as a feature model, implementation artifacts [11, 23, 25] or expert knowledge [7, 9] can be used [30]. However, the majority of sampling algorithms uses a feature model as single input artifact [30]. Hence, our challenge focuses on sampling algorithms using a feature as single input for sampling. Sampling algorithms drastically reduce the number of products to be tested, compared to testing all possible products individually. However, when applied to large product lines from

“A Classification of Product Sampling for Software Product Lines”
Varshosaz et al.
(SPLC 2018)

“Product Sampling for Product Lines: The Scalability Challenge”
Pett et al.
(SPLC 2019)

Golden Set



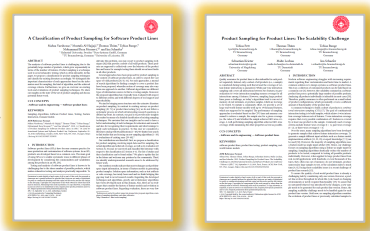
Golden Set



Forward Snowballing



Golden Set



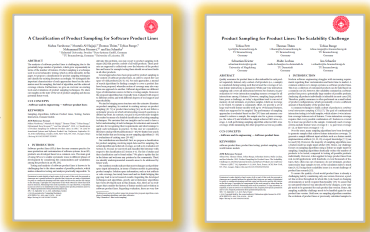
Forward Snowballing



Set of Papers



Golden Set



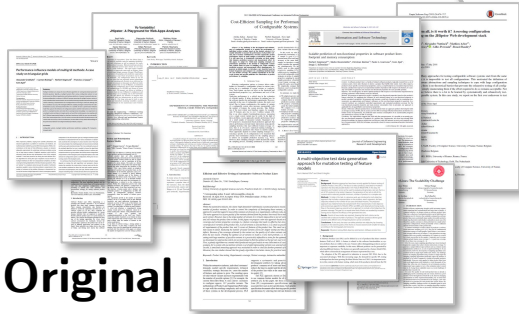
Forward Snowballing



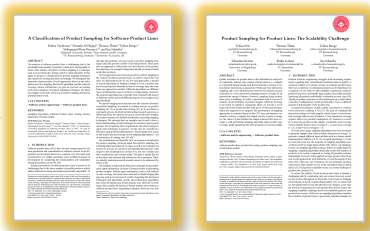
Set of Papers



Papers from Original Survey



Golden Set



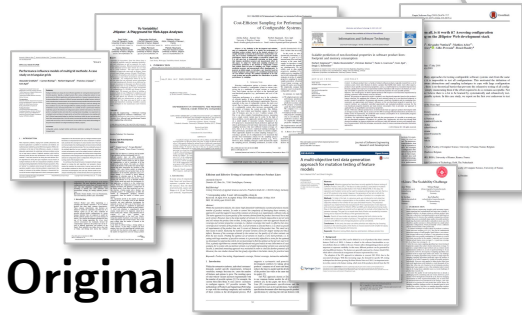
Forward Snowballing



Set of Papers



Papers from Original Survey



Inclusion & Exclusion Criteria

Technical Exclusion

- Conference/ Journal
- English
- Available "for free" online

Content Inclusion

- Introduces, Compares, or Evaluates Sampling Approach(es)
- Topic: Product Sampling or Interaction Testing
- Feature Model, Boolean Algebra, or Decision Model

A Revised Classification of Product Sampling for Software Product Lines – S. Böhm, A. Molt, T.J. Schmidt, T. Thüm | March 23 - March 27 2026

13

Inclusion & Exclusion Criteria

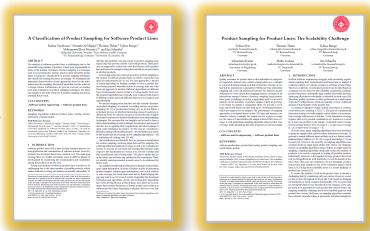
Technical Exclusion

- **Conference/Journal**
- **English**
- **Available “for free” online**

Content Inclusion

- **Introduces, Compares, or Evaluates Sampling Approach(es)**
- **Topic: Product Sampling or Interaction Testing**
- **Feature Model, Boolean Algebra, or Decision Model**

Golden Set



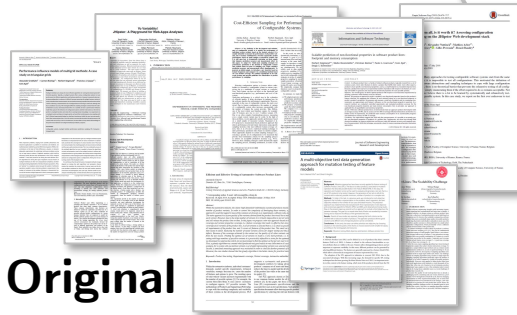
Forward Snowballing



Set of Papers



Papers from Original Survey



Inclusion & Exclusion Criteria

Technical Exclusion

- Conference/ Journal
- English
- Available "for free" online

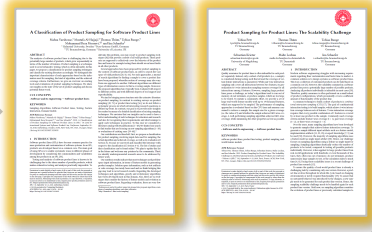
Content Inclusion

- Introduces, Compares, or Evaluates Sampling Approach(es)
- Topic: Product Sampling or Interaction Testing
- Feature Model, Boolean Algebra, or Decision Model

A Revised Classification of Product Sampling for Software Product Lines – S. Böhm, A. Molt, T.J. Schmidt, T. Thüm | March 23 - March 27 2026

13

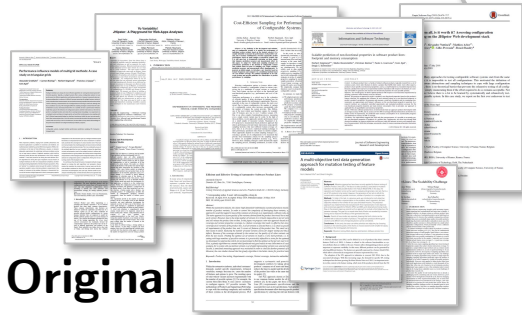
Golden Set



Forward Snowballing



Set of Papers



Papers from Original Survey

Inclusion & Exclusion Criteria

Technical Exclusion

- Conference/ Journal
- English
- Available "for free" online

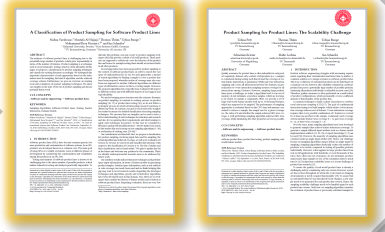
Content Inclusion

- Introduces, Compares, or Evaluates Sampling Approach(es)
- Topic: Product Sampling or Interaction Testing
- Feature Model, Boolean Algebra, or Decision Model

Classification



Golden Set



Forward Snowballing



Set of Papers



Inclusion & Exclusion Criteria

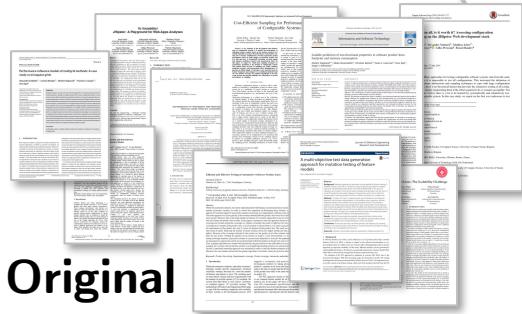
Technical Exclusion

- Conference/ Journal
- English
- Available "for free" online

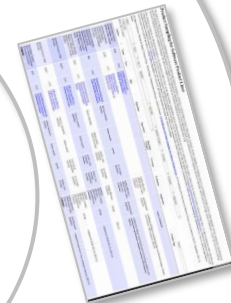
Content Inclusion

- Introduces, Compares, or Evaluates Sampling Approach(es)
- Topic: Product Sampling or Interaction Testing
- Feature Model, Boolean Algebra, or Decision Model

Papers from Original Survey



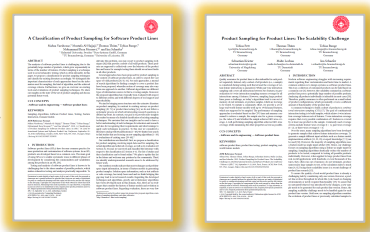
Classification



Forward and Backward Snowballing



Golden Set



Forward Snowballing



Set of Papers



Inclusion & Exclusion Criteria

Technical Exclusion

- Conference/ Journal
- English
- Available "for free" online

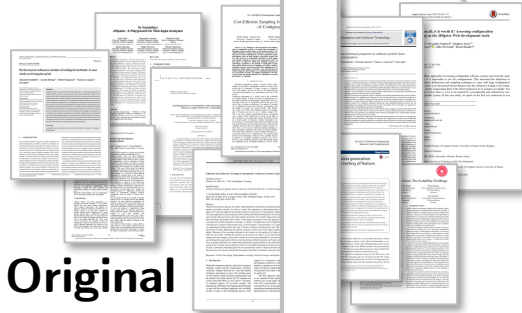
Content Inclusion

- Introduces, Compares, or Evaluates Sampling Approach(es)
- Topic: Product Sampling or Interaction Testing
- Feature Model, Boolean Algebra, or Decision Model

A Revised Classification of Product Sampling for Software Product Lines – S. Böhm, A. Molt, T.J.Schmidt, T. Thüm | March 23 - March 27 2026

13

Papers from Original Survey



Classification



Forward and Backward Snowballing

IEEE Xplore® Springer Google Scholar

ACM DIGITAL LIBRARY Scopus WILEY Online Library

Golden Set

Forward Snowballing

Set of Papers

Inclusion & Exclusion Criteria

Technical Exclusion

- Conference/ Journal
- English
- Available "for free" online

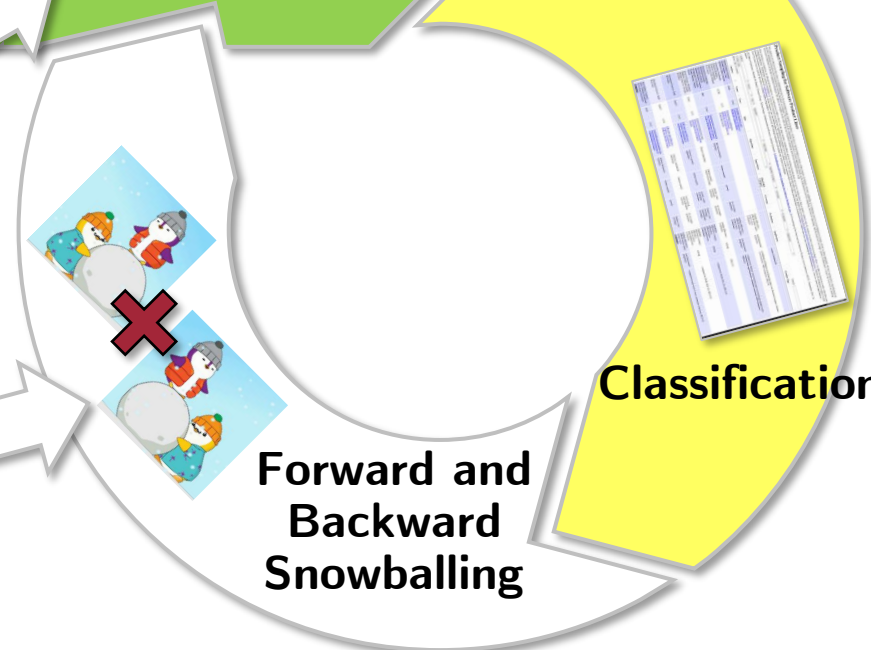
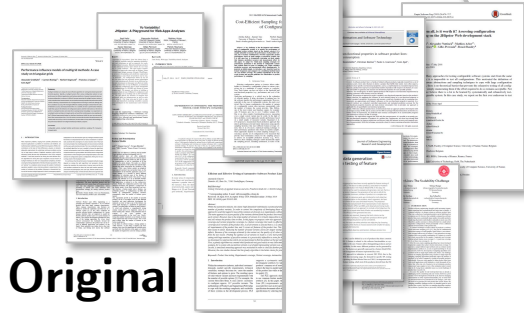
Content Inclusion

- Introduces, Compares, or Evaluates Sampling Approach(es)
- Topic: Product Sampling or Interaction Testing
- Feature Model, Boolean Algebra, or Decision Model

A Revised Classification of Product Sampling for Software Product Lines – S. Böhm, A. Molt, T.J.Schmidt, T. Thüm | March 23 - March 27 2026



Papers from Original Survey



IEEE Xplore®  Springer  Google Scholar
 ACM  DIGITAL LIBRARY  Scopus  WILEY Online Library

Golden Set

Forward Snowballing

Set of Papers

Inclusion & Exclusion Criteria

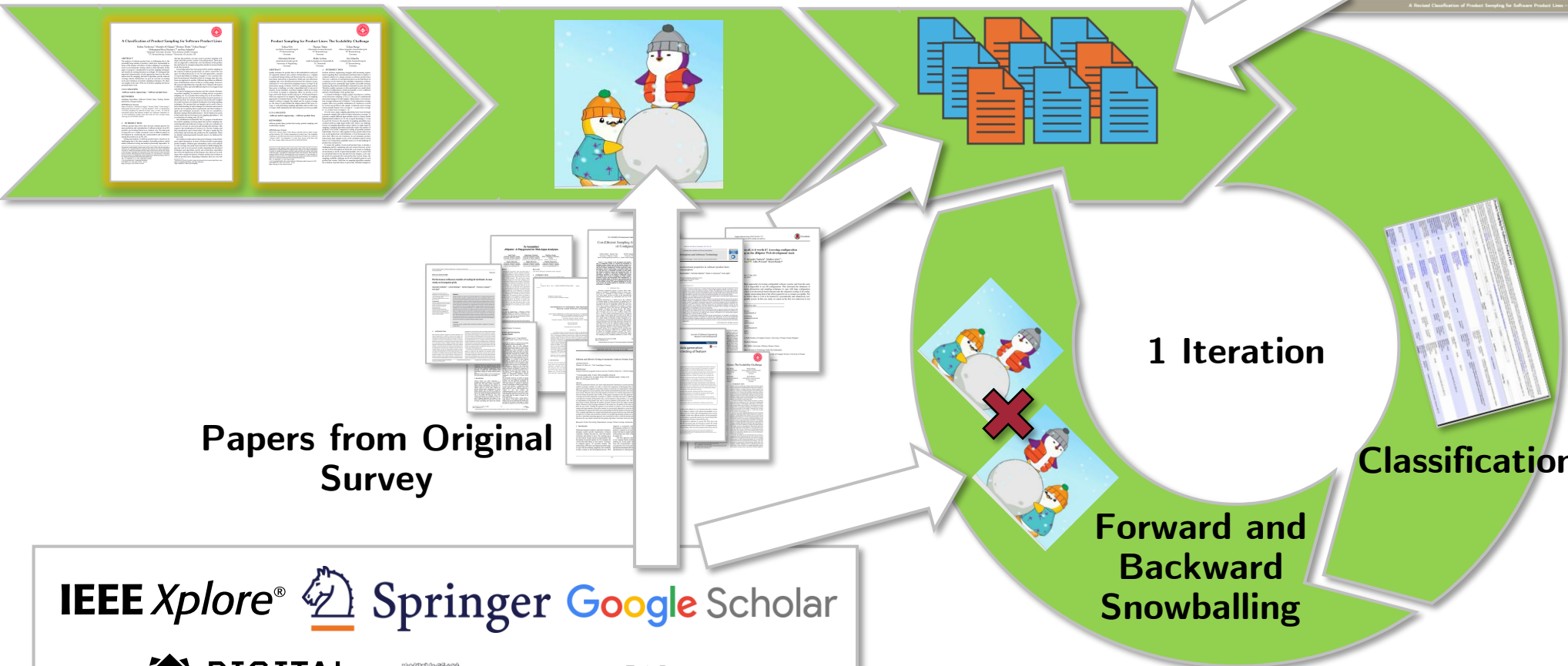
Technical Exclusion

- Conference/ Journal
- English
- Available "for free" online

Content Inclusion

- Introduces, Compares, or Evaluates Sampling Approach(es)
- Topic: Product Sampling or Interaction Testing
- Feature Model, Boolean Algebra, or Decision Model

A Revised Classification of Product Sampling for Software Product Lines – S. Böhm, A. Molt, T.J. Schmidt, T. Thüm | March 23 - March 27 2026



Papers from Original Survey

Forward and Backward Snowballing

Classification

IEEE Xplore®  Springer  Google Scholar
 ACM  DIGITAL LIBRARY  Scopus  WILEY Online Library

Golden Set

Forward Snowballing

Set of Papers

Inclusion & Exclusion Criteria

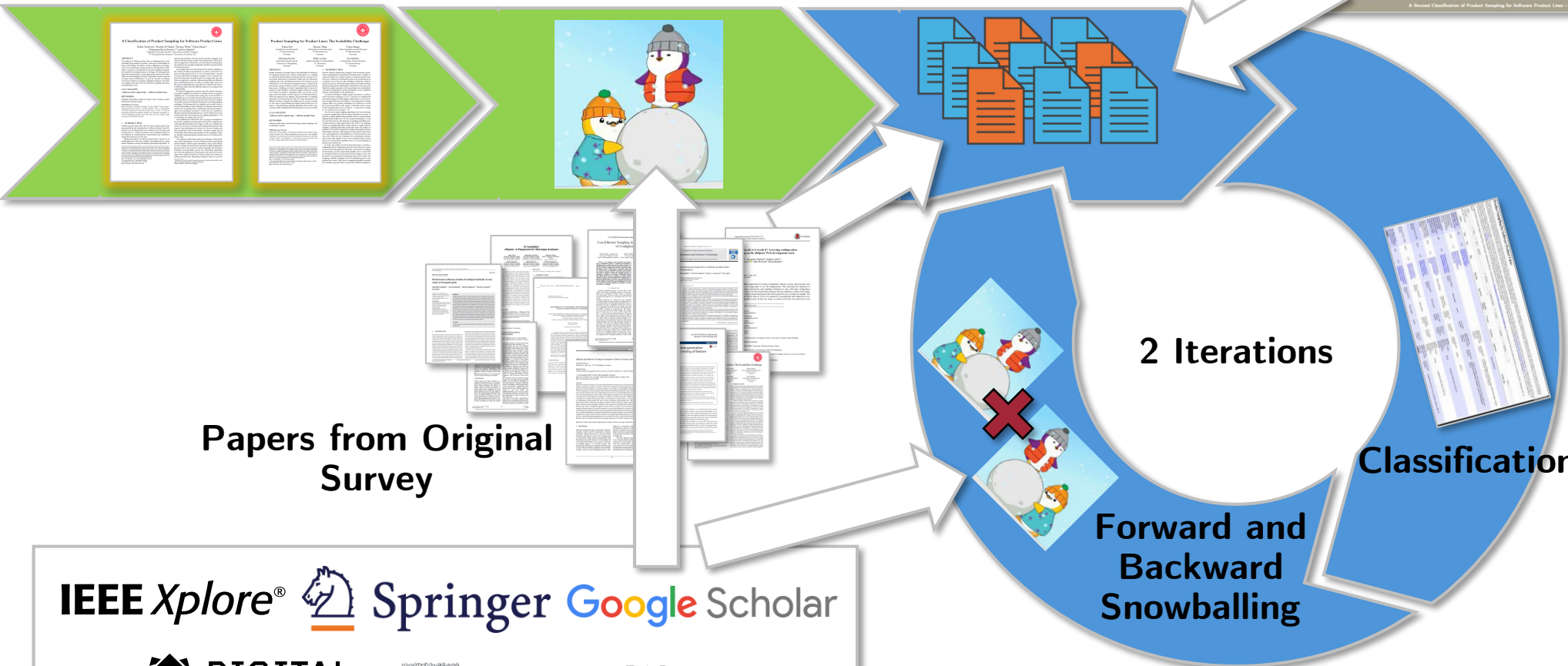
Technical Exclusion

- Conference/ Journal
- English
- Available "for free" online

Content Inclusion

- Introduces, Compares, or Evaluates Sampling Approach(es)
- Topic: Product Sampling or Interaction Testing
- Feature Model, Boolean Algebra, or Decision Model

A Revised Classification of Product Sampling for Software Product Lines - S. Böhm, A. Molt, T.J. Schmidt, T. Thüm | March 23 - March 27 2026



Papers from Original Survey

2 Iterations

Classification

Forward and Backward Snowballing

IEEE Xplore®  Springer  Google Scholar
 ACM  DIGITAL LIBRARY  Scopus  WILEY Online Library

Golden Set

Forward Snowballing

Set of Papers

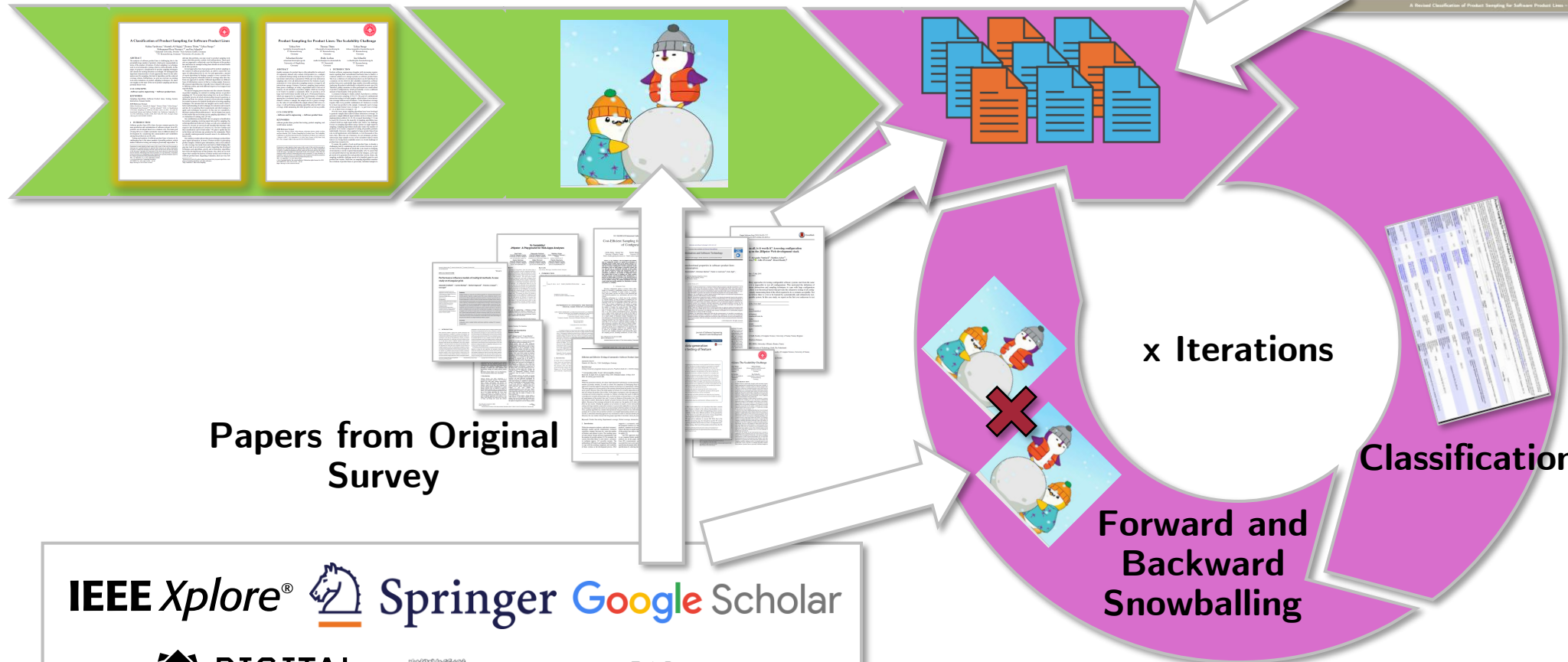
Inclusion & Exclusion Criteria

Technical Exclusion

- Conference / Journal
- English
- Available "for free" online

Content Inclusion

- Introduces, Compares, or Evaluates Sampling Approach(es)
- Topic: Product Sampling or Interaction Testing
- Feature Model, Boolean Algebra, or Decision Model



Papers from Original Survey

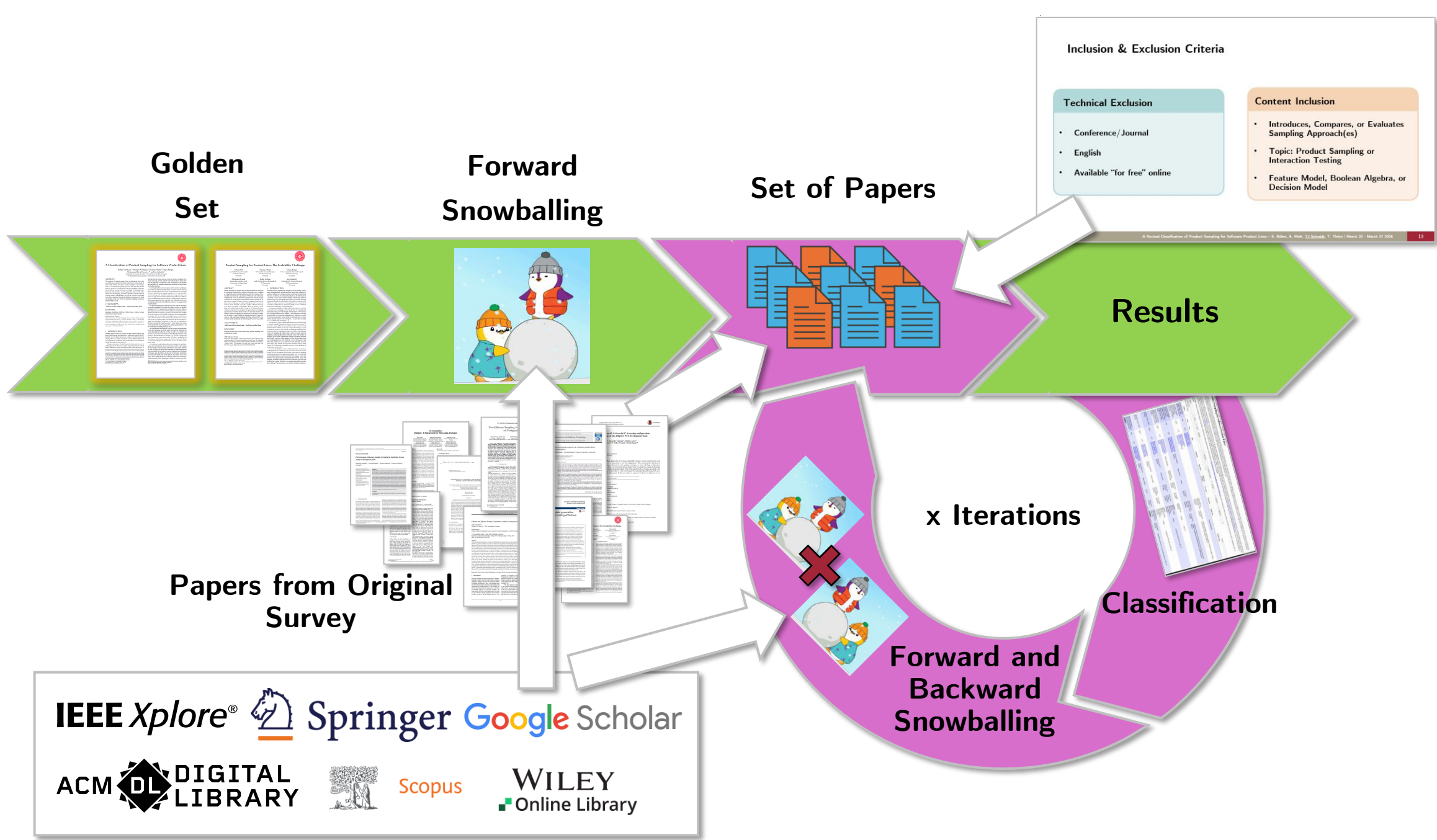
x Iterations

Classification

Forward and Backward Snowballing

IEEE Xplore® Springer Google Scholar

ACM DIGITAL LIBRARY Scopus WILEY Online Library



IEEE Xplore®  Springer  Google Scholar
 ACM  DIGITAL LIBRARY  Scopus  WILEY Online Library

Exact Procedures for t-wise Sampling

How Low Can We Go? Minimizing Interaction Samples for Configurable Systems

DOMINIK M. KRUPKE, AHMAD MORADI, MICHAEL PERK, and PHILLIP KELDENICH,

Algorithms Division, TU Braunschweig, Braunschweig, Germany

GABRIEL GEHRKE, TU Braunschweig, Braunschweig, Germany

SEBASTIAN KRIETER and THOMAS THÜM, Institute of Software Engineering and Automotive

Informatics (ISF), TU Braunschweig, Braunschweig, Germany

SÁNDOR P. FEKETE

Efficient Heuristics and Exact Methods for Pairwise Interaction Sampling:

Sándor P. Fekete[†]

Phillip Keldenich[†]

Dominik Krupke[†]

Michael Perk[†]

Measure quality via lower bounds

- Identify the exact optimal sample when possible + proof
- If exact optimum is unknown, provide upper and lower bounds
- Estimate distance from the optimum using these bounds

Algorithms for Specific Hardware

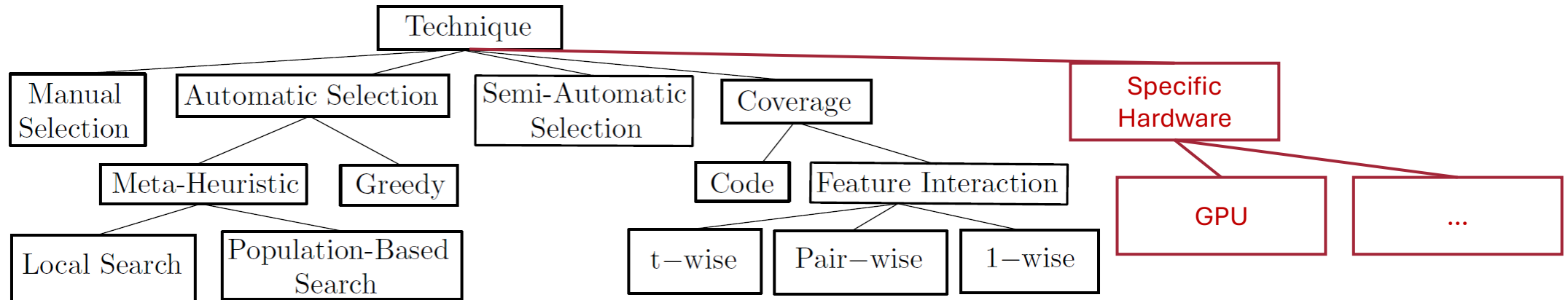
Speed Things Up: Leveraging GPU Processing Capabilities for Faster Sampling of Software Product Lines

Lennard Hettich
University of Stuttgart
Stuttgart, Germany
lennard.hettich@gmx.de

Nasser Jazdi
University of Stuttgart
Stuttgart, Germany
Nasser.Jazdi@ias.uni-stuttgart.de

Johannes Stümpfle
University of Stuttgart
Stuttgart, Germany
johannes.stuempfle@ias.uni-stuttgart.de

Michael Weyrich
University of Stuttgart
Stuttgart, Germany
michael.weyrich@ias.uni-stuttgart.de



Deep Reinforcement Learning Approach to Sampling

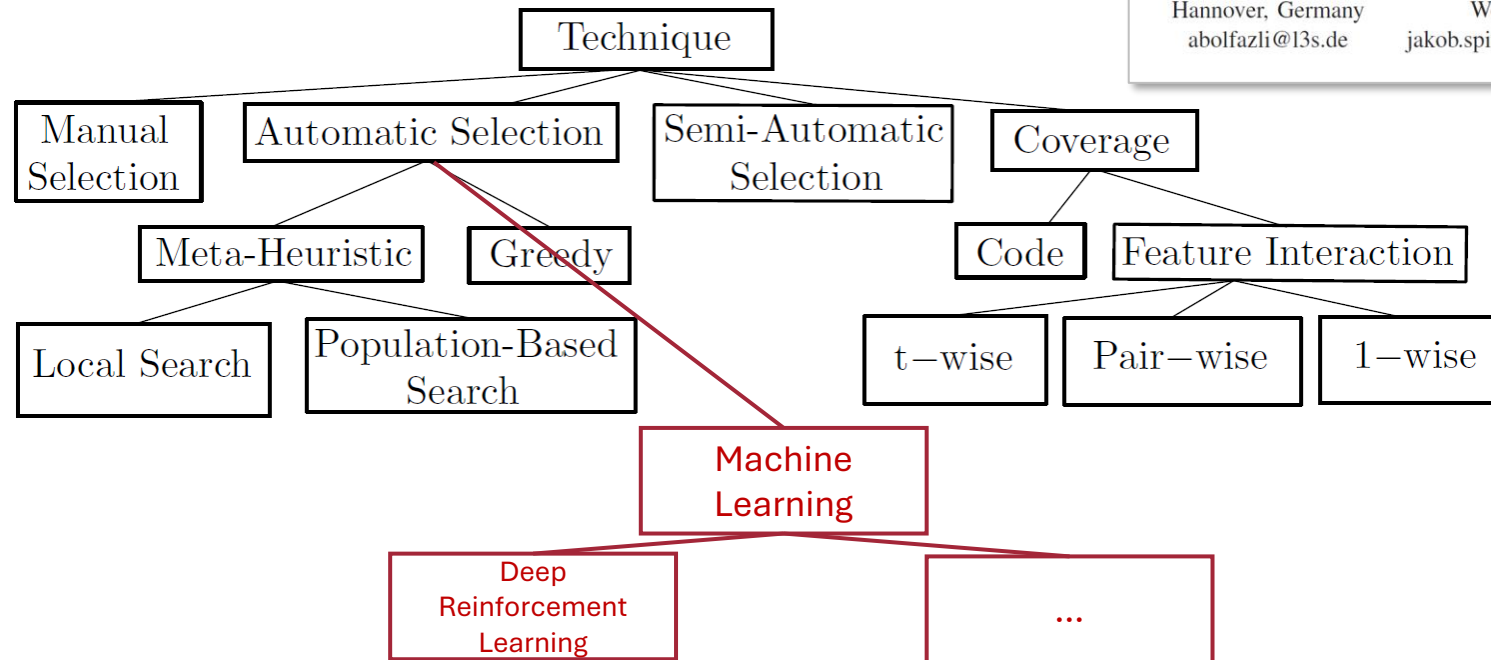
A Deep Reinforcement Learning Approach to Configuration Sampling Problem

Amir Abolfazli
L3S Research Center
Hannover, Germany
abolfazli@l3s.de

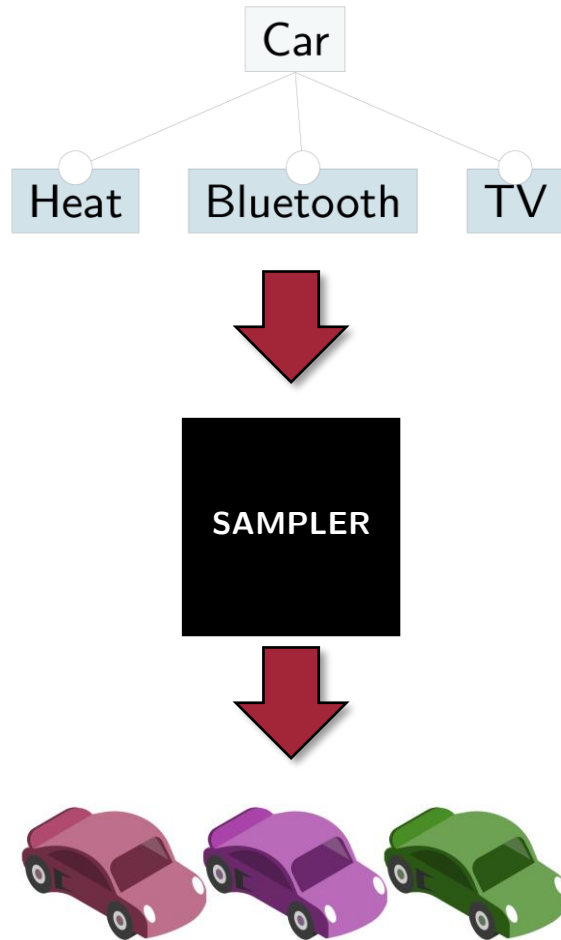
Jakob Spiegelberg
Volkswagen AG
Wolfsburg, Germany
jakob.spiegelberg@volkswagen.de

Gregory Palmer
L3S Research Center
Hannover, Germany
gpalmer@l3s.de

Avishek Anand
Delft University of Technology
Delft, Netherlands
avishek.anand@tudelft.nl



Sampling of Deployed Configurations



Covering T-Wise Interactions of Deployed Configurations

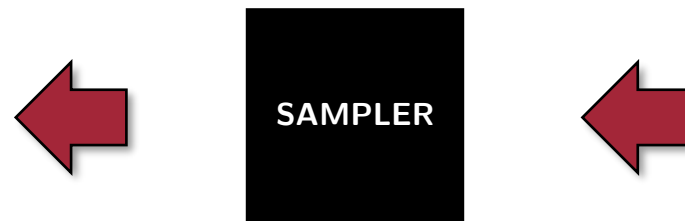
Rahel Sundermann
University of Ulm
Ulm, Germany
rahel.sundermann@uni-ulm.de

Sabrina Böhm
University of Ulm
Ulm, Germany
sabrina.boehm@uni-ulm.de

Sebastian Krieter
TU Braunschweig, Germany
Braunschweig, Germany
sebastian.krieter@tu-
braunschweig.de

Malte Lochau
University of Siegen
Siegen, Germany
malte.lochau@uni-siegen.de

Thomas Thüm
TU Braunschweig, Germany
Braunschweig, Germany
thomas.thuem@tu-braunschweig.de



Iterative Sampling with Additional Analysis

Incremental Identification of T-Wise Feature Interactions

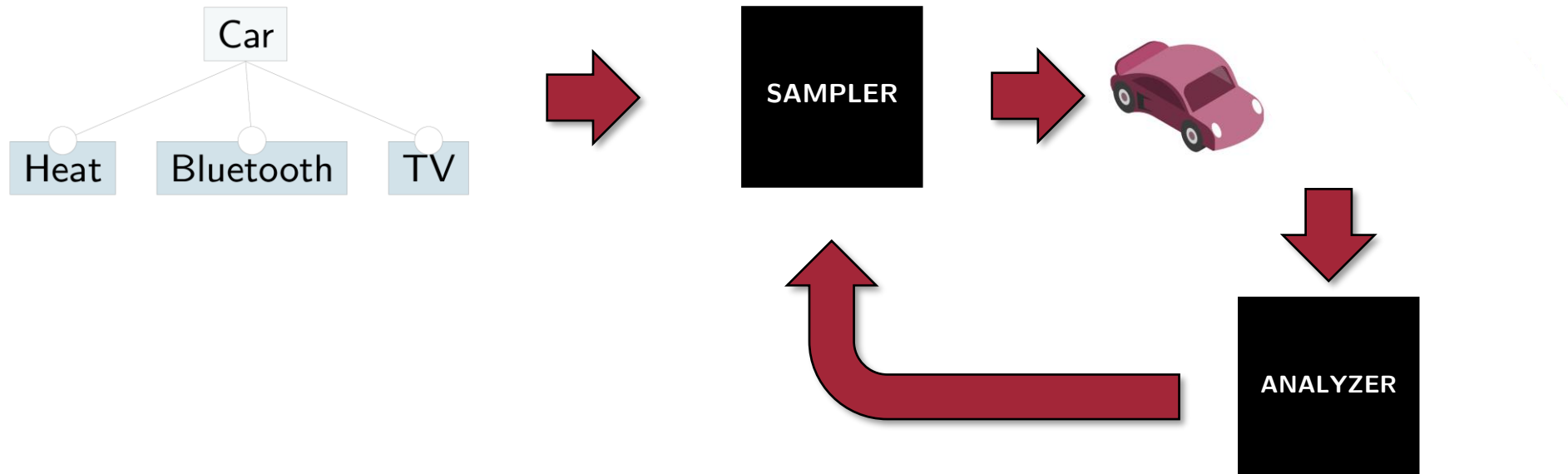
Sabrina Böhm
University of Ulm
Germany

Sebastian Krieter
University of Ulm
Germany

Tobias Heß
University of Ulm
Germany

Thomas Thüm
University of Ulm
Germany

Malte Lochau
University of Siegen
Germany



Iterative Sampling with Additional Analysis

Incremental Identification of T-Wise Feature Interactions

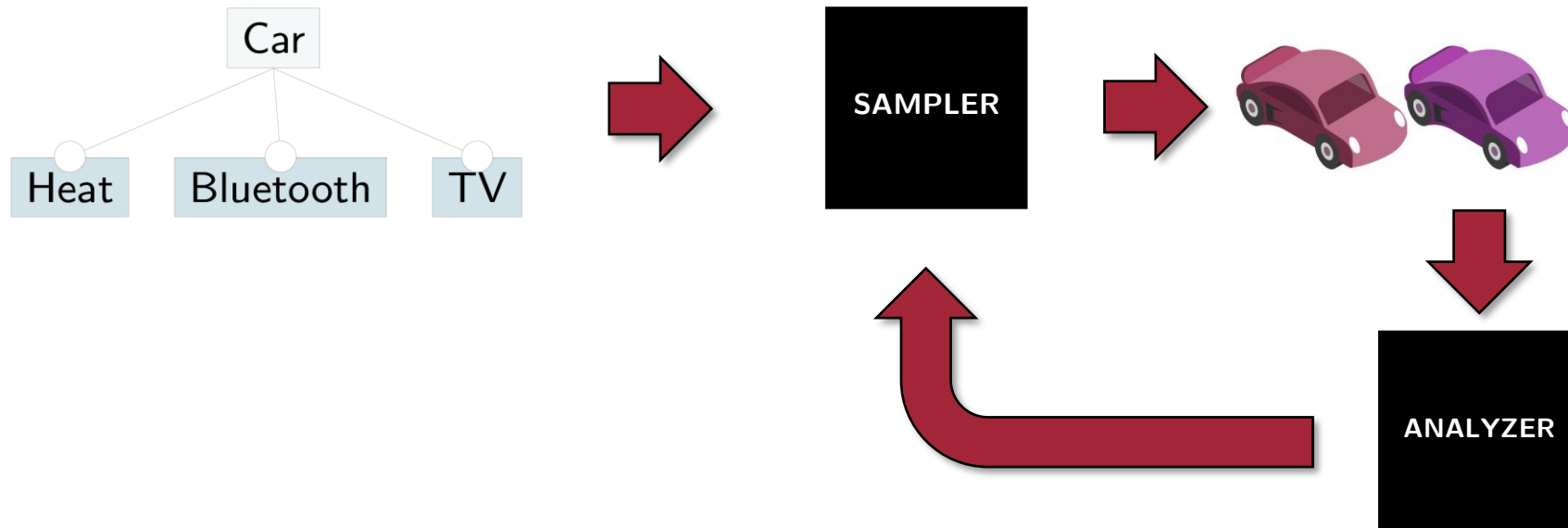
Sabrina Böhm
University of Ulm
Germany

Sebastian Krieter
University of Ulm
Germany

Tobias Heß
University of Ulm
Germany

Thomas Thüm
University of Ulm
Germany

Malte Lochau
University of Siegen
Germany



Iterative Sampling with Additional Analysis

Incremental Identification of T-Wise Feature Interactions

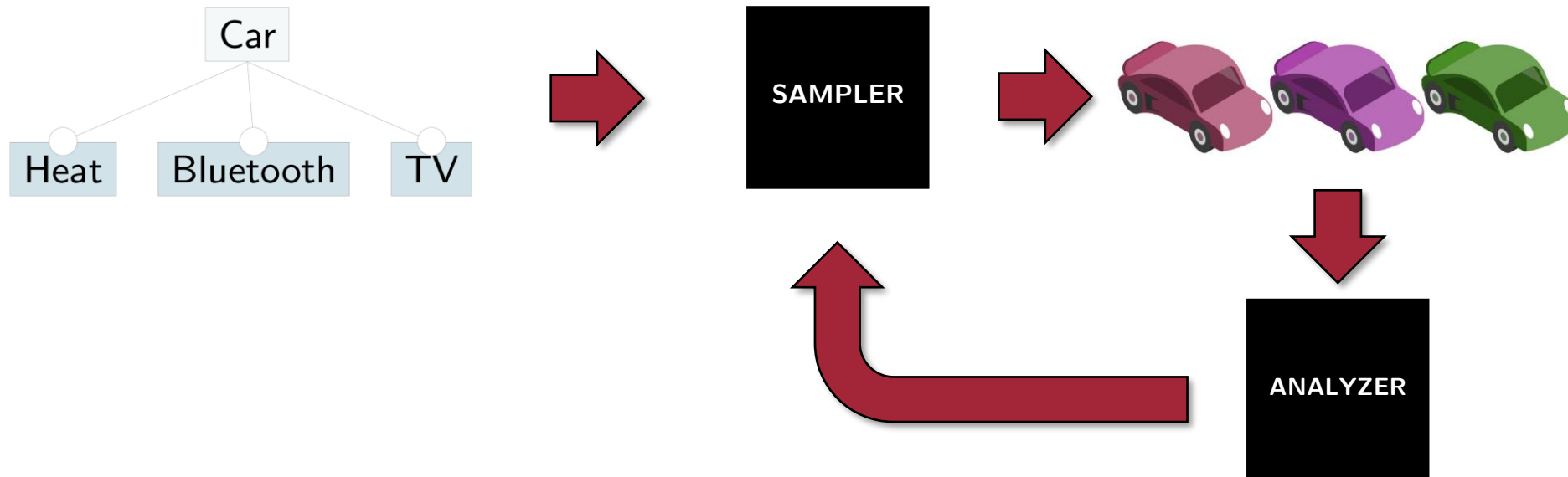
Sabrina Böhm
University of Ulm
Germany

Sebastian Krieter
University of Ulm
Germany

Tobias Heß
University of Ulm
Germany

Thomas Thüm
University of Ulm
Germany

Malte Lochau
University of Siegen
Germany



Maximum Coverage for k Configurations

Solving the t -Wise Coverage Maximum Problem via Effective and Efficient Local Search-Based Sampling

CHUAN LUO and JIANPING SONG, School of Software, Beihang University, Beijing, China

QIYUAN ZHAO, School of Computing, National University of Singapore, Singapore, Singapore

BINQI SUN, Chair of Cyber-Physical Systems in Production Engineering, Technical University of Munich, Munich, Germany

JUNJIE CHEN, College of Intelligence and Computing, Tianjin University, Tianjin, China

HONGYU ZHANG, School of Big Data and Software Engineering, Chongqing University, Chongqing, China

JINKUN LIN, SeedMath Technology Limited, Beijing, China

CHUNMING HU, School of Software, Beihang University, Beijing, China

Idea

Get the highest coverage for a given sample size k

Note

Not seen so far: “for $p\%$ coverage, how many configurations?”

t-wise ≠ t-wise

Coverage Metrics for T-Wise Feature Interactions

Sabrina Böhm
University of Ulm, Germany
0009-0003-4605-8574

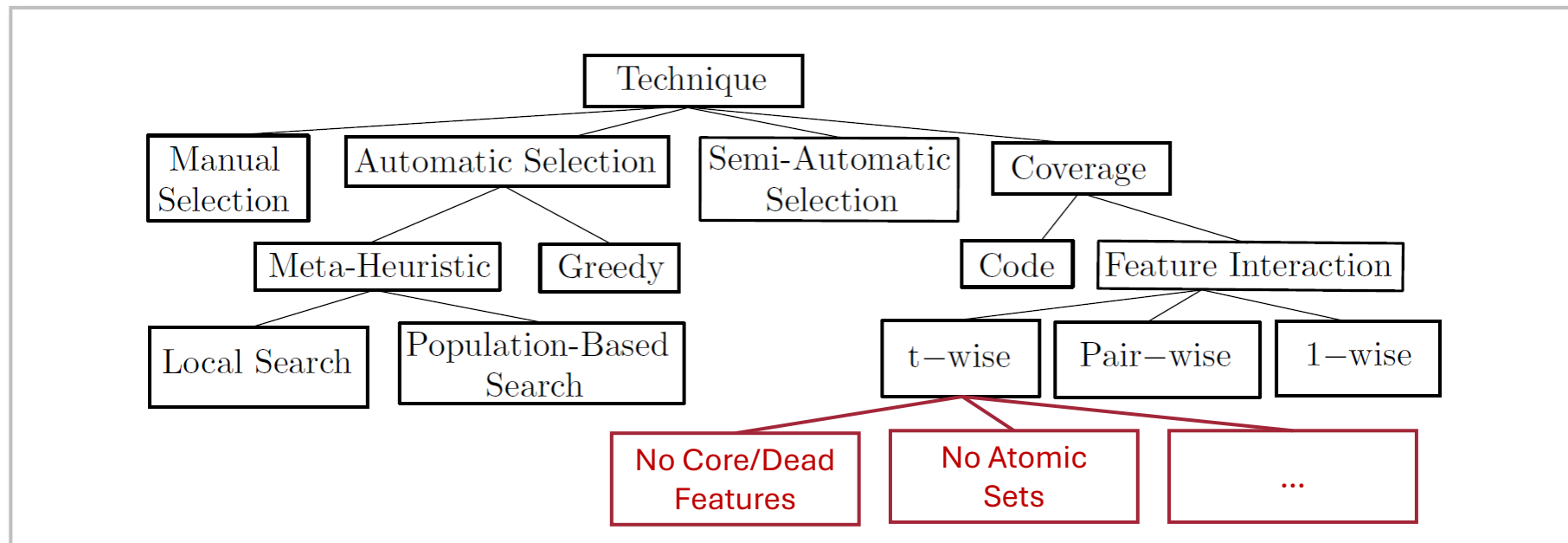
Tim Jannik Schmidt
University of Ulm, Paderborn University, Germany
0009-0005-1650-9500

Sebastian Krieter
TU Braunschweig, Germany
0000-0001-8069-9584

Tobias Pett
Karlsruhe Institute for Technology, Germany
0000-0001-7652-6525

Thomas Thüm
TU Braunschweig, Germany
0000-0001-8069-9584

Malte Lochau
University of Siegen, Germany
0000-0002-8404-753X



A Revised Classification of Product Sampling for Software Product Lines

Institute of Software Engineering and Automotive Informatics
Product Sampling for Software Product Lines

The analysis of software product lines is challenging due to the potentially large number of products, which grow exponentially in terms of the number of features. Product sampling is a technique used to avoid exhaustive testing, which is often infeasible. We have proposed a classification for product sampling techniques and classified the existing literature accordingly. We distinguish the important characteristics of such approaches based on the information used for sampling, the kind of algorithm, and the achieved coverage criteria. Furthermore, we give an overview on existing tools and evaluations of product sampling techniques. We share our insights on the state-of-the-art of product sampling and discuss potential future work.

This website is accepted as an official ACM artifact on our publication at SPLC18 [1]. As the large number of algorithms and our detailed classification give rise to a large space, that is hard to explore only by means of a table in the proceedings. Here, we offer an interactive table with filter and sorting capabilities. Furthermore, product sampling is a very active research area and new algorithms are published several times a year. We aim to keep the table up-to-date, but kindly ask you to [report any missing literature and wrong classifications to us](#). Instead of sending an e-mail it is also possible to propose changes by means of a [pull request](#). In the request, you can add the paper to the file of Bibtex entries, which is used to generate this website. Please note that there is a dedicated entry called `sampling-tags` containing the classification (e.g. `feature-model` as input data or `naae-newAlgorithm` to specify the entry in column entitled Algorithm). We will review the changes manually and update the website accordingly.

[1] Mahsa Varshosaz, Mustafa Al-Hajaji, Thomas Thüm, Tobias Runge, Mohammadreza Mousavi, and Ina Schaefer. [A Classification of Product Sampling for Software Product Lines](#). In *Proceedings of the International Software Product Line Conference (SPLC)*, September 2018. To appear.

Active Filters: No filters applied

Show All entries

Authors	Venue	Year	Title	Algorithm	Input Data	Algorithm Category	Coverage	Evaluation	Application	Further Tags
Abdel Salam Sayyad, Joseph Ingram, Tim Menzies, Harry Ammar	ASE	2013	Scalable Product Line Configuration: A Stear to Break the Camel's Back							usage of JMetal/Z3
Alireza Ensan, Ebrahim Bagheri, Mohsen Asadi, Dragan Gasevic, Yevgen Biletskiy	ITNG	2011	Goal-Oriented Test Case Selection and Prioritization for Product Line Feature Models	Ensan's algorithm	feature model, expert knowledge	greedy, semi-automatic selection	no coverage guarantee	testing efficiency, evaluation	testing	SPLC18
Anastasia Cmyrev, Ralf Reising	IAST	2014	Efficient and Effective Testing of Automotive Software Product Lines	Cmyrev's greedy algorithm	feature model	greedy, semi-automatic selection	feature-wise coverage, requirements coverage	sampling efficiency, testing efficiency, effectiveness, unavailable tool, evaluation	testing	compared to MoSo-PoliTe, SPLC18
Anastasia Cmyrev, Ralf Reising	IAST	2014	Efficient and Effective Testing of Automotive	Cmyrev's simulated annealing	feature model	local search, semi-automatic selection	no coverage guarantee	sampling efficiency, testing efficiency	testing	compared to MoSo-PoliTe, SPLC18

<https://tubs-isf.github.io/>

Question for you:

Are we missing recent developments or further categories?

Discoveries

- Exact Procedures for t-wise Sampling
- Algorithms for Specific Hardware
- Deep Reinforcement Learning Approach to Sampling
- Sampling of Deployed Configurations
- Iterative Sampling with Additional Analysis
- Maximum Coverage for k Configurations
- t-wise \neq t-wise