Bachelor's Thesis

The Impact of Coverage Metrics on the Evaluation of Sampling Strategies in Configurable Systems

Michael Hendrik Echternach

July 2, 2026

Advisor:

Tobias Dick	Chair of Software Engineering
Kallistos Weis	Chair of Software Engineering

Examiners:

Prof. Dr. Sven Apel	Chair of Software Engineering
Prof. Dr. Martina Maggio	Chair of Software and Hardware Systems

Chair of Software Engineering
Saarland Informatics Campus
Saarland University

UNIVERSITÄT
DES
SAARLANDES

Erklärung
Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne die Beteiligung dritter Personen verfasst habe, und dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen wurden, sind als solche kenntlich gemacht. Insbesondere bestätige ich hiermit, dass ich bei der Erstellung der nachfolgenden Arbeit mittels künstlicher Intelligenz betriebene Software (z. B. ChatGPT) ausschließlich für folgende zulässige Teilaufgaben: Überarbeitung/Anpassung von Texten, Überprüfung auf Fehler wie falsche Kommasetzung und Rechtschreibfehler, Brainstorming, Code-Vervollständigung und nicht zur Bearbeitung der in der Arbeit aufgeworfenen Fragestellungen zu Hilfe genommen habe. Mir ist bewusst, dass der Verstoß gegen diese Versicherung zum Nichtbestehen der Prüfung bis hin zum Verlust des Prüfungsanspruchs führen kann.

Ich erkläre mich damit einverstanden, dass die Arbeit mittels eines Plagiatsprogrammes auf die Nutzung einer solchen Software überprüft wird.

I hereby declare that this thesis is my own original work and was completed independently, without unauthorized assistance or unacknowledged sources. Any content derived from publications or other external sources, whether quoted verbatim or paraphrased, has been duly acknowledged and clearly marked as such. I hereby confirm that, in preparing the following thesis, I have used artificial intelligence-based software (such as ChatGPT) solely for the following permitted tasks: text rewriting/revision, check for mistakes like comma placements and spelling errors, brainstorming, code completion, and finding of bugs, and not to address the core research questions presented in this thesis. I acknowledge that any breach of this declaration may result in failing the examination and, in severe cases, the right to be examined may be revoked.

I consent that my thesis may be checked using plagiarism and AI detection software.

Saarbrücken, 03.07.2026 _____

(Datum Date) (Unterschrift Signature)

Einverständniserklärung (optional)
Declaration of Consent (optional)

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

I consent to having both the printed and digital versions of my thesis, which has been examined and awarded at least a passing grade, added to the Computer Science Department's library and thus made publicly accessible.

Saarbrücken, 03.07.2026 _____

(Datum Date) (Unterschrift Signature)

Abstract

Many software systems are developed as configurable systems to support variability across different use cases, platforms, and performance requirements by selecting combinations of features. However, increasing the number of features causes the number of possible combinations and resulting configurations to grow exponentially. Evaluating every configuration therefore becomes computationally infeasible. Consequently, researchers and practitioners rely on sampling strategies to select a manageable subset of configurations for testing, analysis, or learning tasks. Different sampling strategies exhibit different strengths and trade-offs.

However, recent work has shown that there is no consistency on how to compute the coverage of feature combinations of size t when constraints between features restrict the valid configurations. Multiple coverage metrics have been identified in the literature, each considering different subsets of feature interactions. As a result, the reported coverage of the same sample may vary depending on the chosen metric. This can lead to inconsistent conclusions about the effectiveness of sampling strategies.

In this thesis, we systematically use eight coverage metrics by combining the interaction filters identified in prior work. We use them to evaluate common sampling strategies. Our empirical study analyses samples produced by random, distance-based, and solver-based sampling, using t-wise sampling as a reference for determining the sample sizes. The evaluation is conducted on 21 real-world configurable software systems. Our analysis investigates whether previously reported conclusions about the interaction coverage achieved by sampling strategies remain stable across different coverage metrics.

By systematically evaluating sampling strategies across multiple coverage metrics, this thesis demonstrates how the choice of metric influences the assessment and comparison of sampling strategies in configurable systems.

Acknowledgments

First of all, I would like to express my deepest gratitude to my supervisors, Kallistos Weis and Tobias Dick, who supported me throughout my work on this thesis, provided me with valuable feedback, and consistently encouraged me.

I would also like to thank Professor Sven Apel and Professor Martina Maggio for reviewing my bachelor's thesis.

My special thanks also go to Nils Husung, who spent countless hours helping us with the BDD implementation. I am also very grateful to Qingqing Dong for her implementation of the distance-based and solver-based sampling methods.

I would like to thank Shao-Yun Guo and, in particular, Marcel Niedballa for their guidance, help, and support.

Finally, I would like to thank everyone who supported me throughout the entire period during which I worked on this thesis.

Contents

List of Figures

List of Tables

1 Introduction

Configurable systems provide multiple features, which are not only used for adapting functionality, but also non-functional properties like performance and energy consumption [67]. Within a software product line (SPL), features reflect the requirements of stakeholders and describe how the individual variants differ from one another [4, 40]. A *configuration* describes a single combination of features [15]. These features come with their *dependencies* (e.g., exclusions) and are often represented in so-called *feature models*, which describe valid combinations of features [11, 13, 38].

This can result in a combinatorial explosion of the configuration space [15, 32]: as the number of features n increases, the number of configurations and potential interactions grows exponentially (2^n). The result is that a full measurement of the performance of each individual variant is no longer computationally feasible, even for medium-sized systems, let alone larger ones. The Linux kernel, for example, contains more than a thousand features [32], resulting in a configuration space that is far too large to test every interaction individually. In practice, this problem is compounded by the fact that configurable systems often offer far more features than are actually necessary [70].

To overcome this limitation, researchers employ machine learning to infer *Performance-Influence Models* based on a limited budget of measurements (i.e., training set) [58]. However, the accuracy of these predictions is greatly dependent on the quality of the training set, specifically, whether the selected configurations adequately cover the feature interactions [37].

To efficiently select configurations, there are many *sampling strategies*. *Uniform random sampling* selects configurations stochastically without targeting specific structural coverage goals. While it ensures that the final selection consists only of valid configurations, finding these valid samples is computationally challenging in highly constrained environments [33]. *Solver-based sampling* uses a dependency solver to draw configurations and increases the variety of configurations. However, it often happens that it generates clusters instead of achieving uniform distribution [37]. *Distance-based sampling* solves the cluster problem by using both a distance metric and a solver to spread configuration more evenly across the space [37]. In *t-wise sampling*, every possible combination of t features should be included at least once. However, an increase in the interaction strength t ($t \geq 2$) leads to a significant increase in the number of configurations needed for complete coverage. The size of the minimal covering set increases exponentially with t. For large-scale systems, generating dependency-satisfying t-wise samples becomes computationally infeasible, as the required sample size and the cost of analysing each additional configuration grow rapidly with t [7, 46, 50].

To address the prohibitive cost of full t-wise sampling, researchers have studied the effectiveness of partial t-wise sampling [15, 53]. In this approach, a measurement budget is used

to select only a subset of the fully covering design. A key challenge here is to quantify the quality of these subsets. More specifically, the aim is to determine what percentage of the characteristic interactions are actually covered by the partial sample. Consequently, the evaluation of these strategies is entirely dependent on the definition and computation of *partial coverage metrics* [15].

A recent study by Böhm et al. [15] identified a fundamental problem. Although t-wise sampling is widely used, there is no consensus on how to compute the t-wise coverage of a sample w.r.t. the dependencies between features. To address this issue, they conducted a literature review, where they came across at least six different coverage metrics which were used in the research community. Böhm et al. researched the impact of the choice of metric. The result was that for a sample, coverage values can differ by up to 21% depending on the chosen metric, and some metrics require considering only half the feature interactions of others. This discrepancy is critical because it implies that the perceived effectiveness and ranking of sampling strategies may be an artefact of the chosen metric.

This can be seen across several works: Kaltenecker et al. [37], for instance, compared different sampling strategies using a single notion of coverage. On the other hand, Böhm et al. [15] focused on how the different metrics differ, but how these differences can affect the evaluation of sampling techniques themselves was not covered. Without this, we do not know whether different coverage metrics change the quality of sampling strategies.

To address this gap, we derive our own coverage metrics by systematically combining the interaction filters identified by Böhm et al. [15], and use these metrics to evaluate common sampling strategies. We do this by looking at how the coverage of existing sampling strategies, such as random, solver-based, and distance-based sampling differs when applying the metrics. We conduct an evaluation on a set of configurable systems. For each feature model, we consider only valid configurations which are defined by the model's constraints. For each feature model and sampling strategy, we generate a sample of valid configurations with the corresponding sampling strategy. From the generated sample, we derive partial samples of increasing size. Of each partial sample, we compute the coverage values using all the selected coverage metrics. For the randomised sampling strategies, we repeat the procedure with different random seeds. After the evaluation, we analyse the coverage differences by using a consistent evaluation framework. Our analysis shows that the coverage metrics correlate positively but are not numerically interchangeable, that the ranking of sampling strategies remains stable across all metrics, and that the negative relationship between the strength of the system constraints and the coverage is also metric-independent. In other words: the metrics differ in the absolute coverage values they yield, but not in the conclusions to which they lead. This means that, when evaluating sampling strategies, the choice of coverage metric is less important than the conditions under which coverage is measured, in particular the coverage strength and the sample size.

2

Background

This section introduces the basic concepts required to understand this thesis. The section covers configurable software systems and feature models, common sampling strategies for configuration spaces, and coverage metrics for evaluating configuration samples.

2.1 Configurable Software Systems

Software systems are increasingly developed as *configurable software systems* that can be tailored to diverse requirements by choosing various combinations of features [25]. This approach is designed to improve reusability and development productivity [25]. Features represent qualitative properties of concepts [21]. These features involve certain *dependencies* that restrict which feature combinations are valid. These dependencies define the space of permitted system variants. A *feature model* captures these features and their relationships, for example, by defining mandatory, optional, alternative, or grouped features [38].
Feature modelling, originally introduced by Kang et al. [38], has emerged as the standard approach to modelling variability [11, 13].

Definition 1 **(Feature Model).** A feature model is a tuple $M = (F, D)$, where $F = \{f_1, \ldots, f_n\}$ is a set of features and $D = \{d_1, \ldots, d_m\}$ is a set of dependencies over the Boolean literals $L(F) = \{f_1, \ldots, f_n, \neg f_1, \ldots, \neg f_n\}$ [15]. The dependencies are represented as propositional formulas. We denote the universe of all feature models with $\mathcal{M}$.

A *configuration* represents a specific set of features.

Definition 2 **(Configuration).** Let $M = (F, D)$ be our feature model. We define a *configuration* as a total function $c : F \to \{0, 1\}$, which assigns every feature from the feature set F a 0 if it is deselected and 1 if it is selected. We denote by C_M the set of all valid configurations of M, which are the configurations that fulfil all dependencies.

Definition 3 **(Partial Configuration).** Let $M = (F, D)$ be a feature model. We define a *partial configuration* as a partial function $\gamma : F \rightharpoonup \{0, 1\}$, which assigns boolean values to only a subset of F. The *semantics* of γ, which is denoted by $[\![\gamma]\!]$, is defined as the set of all valid configurations that are consistent with γ:

$$[\![\gamma]\!] = \{\, c \in C_M \mid \forall f \in \mathrm{dom}(\gamma) : c(f) = \gamma(f) \,\}.$$

A (partial) configuration is *valid* if the semantics are not empty. Since feature interactions occur in situations where the combined composition of two or more features results in new behaviour [6], they can only become apparent in configurations where the relevant features are selected simultaneously.

Definition 4 (Feature Interaction). Let $M = (F, D)$ be a feature model. A t-wise feature interaction is a partial configuration $i : F \rightharpoonup \{0,1\}$ with $|\mathrm{dom}(i)| = t$. The interaction is valid if $[\![i]\!]_M \neq \emptyset$. We define $\diamondsuit(M,t)$ as the set of all valid t-wise feature interactions of M. Formally,

$$\diamondsuit(M,t) = \{i \colon F \rightharpoonup \{0,1\} \mid |\mathrm{dom}(i)| = t \wedge [\![i]\!]_M \neq \emptyset\}.$$

We denote by I the universe of all feature interactions.

Figure 2.1 shows an example of a feature model. In this model, the features are structured as a tree that defines parent-child relationships. Selecting a feature always requires selecting its parent feature in the feature model. In this model, *Web Application* acts as the parent feature with child features such as *Encryption* and *Standalone*. These relationships are categorised by specific selection rules that determine the configuration of a product. When child features are organised into groups, they follow additional logic: an *Alternative (XOR) Group*, containing *TLS* and *E2E*, requires that exactly one feature be chosen, while an *OR Group*, such as *On-Premise* and *Cloud*, requires the selection of at least one feature. To extend the hierarchical structure of feature models, researchers use *cross-tree constraints* to further improve configuration [12]. An example of this is shown below the tree in Figure 2.1: selecting *Standalone* requires selecting its parent *Web Application*, which is in any case the always-selected root of the model. The cross-tree constraint, however, simultaneously forbids *Web Application* whenever *Standalone* is selected. These two requirements cannot both hold, so *Standalone* can never appear in any complete valid configuration. It is therefore a *dead feature*: a feature that can never be selected. A *mandatory feature*, such as *Encryption*, must always be selected in every complete valid configuration and cannot be deselected, while an *optional feature*, such as *Deployment*, may be included or excluded at the choice of the user.

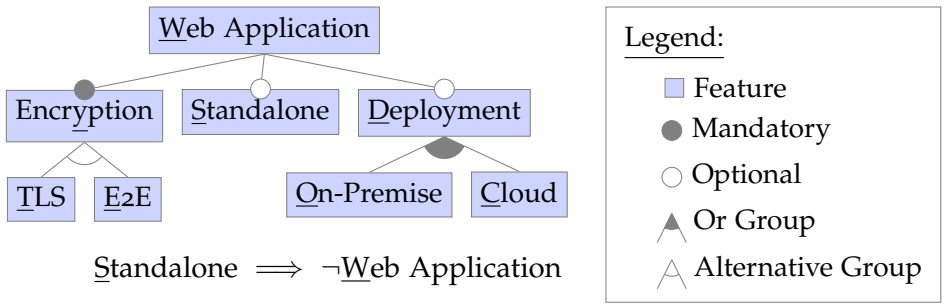


Figure 2.1: A feature model representing a graphical product line

Example 1 **(Feature Model and Configuration).** Based on our feature model in Figure 2.1, our feature model is defined as $M = (F, D)$ where F is defined as $F = \{w, y, t, e, s, d, o, c\}$ and D as $D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$ with
$d_1 : w = \text{true}$, $d_2 : w \iff y$, $d_3 : s \implies \neg w$, $d_4 : d \implies w$, $d_5 : y \iff (t \vee e)$, $d_6 : \neg(t \wedge e)$, $d_7 : d \iff (o \vee c)$
This feature model M gives rise to a set of various configurations. A valid configuration $c_{complete}$ may be defined as:

$$c_{complete}(f) = \begin{cases} 1 & \text{if } f \in \{w, y, e, d, o, c\} \\ 0 & \text{if } f \in \{t, s\} \end{cases}$$

In contrast, a valid partial configuration γ may be defined as:

$$\gamma(w) = 1, \quad \gamma(y) = 1, \quad \gamma(t) = 1, \quad \gamma(s) = 0.$$

A partial configuration $\gamma_{invalid}$ is invalid if $[\![\gamma_{invalid}]\!] = \emptyset$.

2.2 Sampling in Configurable Systems

It is usually infeasible to list all valid configurations of a feature model [46, 64]. Consequently, both researchers and practitioners rely on *sampling strategies* to select a manageable subset of configurations for testing, analysis, or learning tasks.

Random Sampling. Random sampling selects a random subset of the valid configurations, with each valid configuration being equally likely to be selected in the sample [2, 33]. However, implementing pseudo-randomness, which is the only way to implement the sampling strategy on classical computers, results in having a small bias [2]. Many optimisations have been proposed, but these often struggle to scale, for example when they are based on *BDD* representations whose synthesis depends on finding a suitable variable ordering, which is not always feasible [33].

***t*-wise Sampling.** In order to ensure structural coverage of feature interactions, t-wise sampling seeks to cover all valid t-wise feature interactions [15, 35]. For $t = 1$, the goal is to guarantee that every feature is both selected and deselected at least once (feature-wise coverage). For $t = 2$, all valid pairs of features are covered at least once (pairwise coverage) [46, 50]. Due to computational limitations, practical applications usually restrict t to small values, typically $t \leq 3$ [15, 46, 50]. Despite its widespread use, t-wise sampling has significant limitations. Generation algorithms do not scale well for larger values of t or highly constrained models [50]. Furthermore, due to its design, t-wise sampling ignores interactions of order greater than t, which can lead to the exclusion of higher-order effects relevant to system behaviour, including performance anomalies [37].

Solver-Based Sampling. Solver-based sampling uses standard dependency solvers (e.g.,

SAT solvers such as SAT4J [14]) to directly generate valid configurations from a feature model [37]. Typically, the solver is called repeatedly and the first k solutions are collected [18, 37]. Due to solver heuristics, subsequent solutions are often structurally similar, leading to biased samples [31]. Randomised variants address this issue by altering the solver's state between runs, for example by mixing variables or dependencies [31].

Distance-Based Sampling. Unlike sampling strategies that explicitly maximise structural interaction coverage (e.g., t-wise sampling), distance-based sampling gives each valid configuration a *distance value* by using a *distance metric*. This distance value generally reflects the number of configuration options selected and serves as an indicator of the degree of interaction between the features. In addition, distance-based sampling uses a *discrete probability distribution* over the possible distance values in order to control which configurations to select. Configurations are then sampled such that the resulting sample set is spread across the configuration space according to the chosen distribution, rather than focusing on specific interaction orders [37].

2.3 Coverage Metrics

In this section, we introduce the metrics that we use for our t-wise coverage computation. We base our definitions of coverage metrics on the paper *Coverage Metrics for T-Wise Feature Interactions* by Böhm et al. [15]. We include only a subset of the filters, as certain filters require prior knowledge of the feature model. Furthermore, our implementation of the feature model structure diverges from the approach used by Böhm et al.; specifically, they explicitly mark abstract features in their model file, whereas we do not. For a feature model M, an interaction filter ∇, a strength $t \in \mathbb{N}$, and a sample $S \subseteq C_M$, the *t-wise coverage* is defined as:

$$Cov(\nabla, t, M, S) = \begin{cases} 1, & \text{if } \nabla(\diamondsuit(M,t)) = \emptyset, \\ \dfrac{|\{i \in \nabla(\diamondsuit(M,t)) \mid \exists c \in S : c \in [\![i]\!]_M\}|}{|\nabla(\diamondsuit(M,t))|}, & \text{otherwise.} \end{cases}$$

$\nabla : I \to I$ defines our filter, which disregards certain interactions in the computation. For readability, we omit the feature model M in our definition. In our context, however, the M feature model is fixed. With these definitions, we present the filters which exclude or merge subsets of features. These filters form the basis for our coverage metrics:

No filter (Default): this filter is the identity function and describes the default coverage metric $Cov_{default}$:

$$\nabla_{default}(I) = I$$

Excluding Mandatory Features (MF): this filter omits all mandatory features from the model.

Table 2.1: The eight metrics used in the evaluation and their corresponding filters

Metric	MF	DF	ALS	PCI	Formula
MDAP	✓	✓	✓	✓	$\nabla_{MF-DF-ALS-PCI} = \nabla_{MF} \circ \nabla_{DF} \circ \nabla_{ALS} \circ \nabla_{PCI}$
MDA	✓	✓	✓		$\nabla_{MF-DF-ALS} = \nabla_{MF} \circ \nabla_{DF} \circ \nabla_{ALS}$
MDP	✓	✓		✓	$\nabla_{MF-DF-PCI} = \nabla_{MF} \circ \nabla_{DF} \circ \nabla_{PCI}$
MD	✓	✓			$\nabla_{MF-DF} = \nabla_{MF} \circ \nabla_{DF}$
AP			✓	✓	$\nabla_{ALS-PCI} = \nabla_{ALS} \circ \nabla_{PCI}$
PCI				✓	∇_{PCI}
ALS			✓		∇_{ALS}
Default					$\nabla_{default}$

For instance, in Figure 2.1, both *Web Application* and *Encryption* would be excluded. We use the following definition for the corresponding filter ∇_{MF}:

$$\nabla_{MF}(I) = \{i \in I \mid \text{dom}(i) \cap l_{mandatory(M)} = \emptyset\}$$

where $l_{mandatory(M)}$ is the set of all mandatory features of a feature model M.

Excluding Dead Features (DF): this filter removes all dead features. In the case of Figure 2.1, *Standalone* would be filtered out. We use the following definition for the corresponding filter ∇_{DF}:

$$\nabla_{DF}(I) = \{i \in I \mid \text{dom}(i) \cap l_{dead(M)} = \emptyset\}$$

where $l_{dead(M)}$ is the set of all dead features of a feature model M.

Merging Atomic Literal Sets (ALS): this filter merges atomic literal sets, which are a generalisation of atomic feature sets. Features within the *atomic feature set* always occur together in every valid configuration [15]. An atomic set of literals contains all literals (positive and negative) that must be present in a valid configuration if even one of the literals from that set is present in the configuration. We use the following definition for the corresponding filter ∇_{ALS}:

$$\nabla_{ALS}(I) = \{i \in I : \forall a \in l_{atomicLiteralSets(M)} : |a| > 1 \Rightarrow \text{dom}(i) \cap (a \setminus \{\Pi_1(a)\}) = \emptyset\}$$

where $l_{atomicLiteralSets(M)}$ is the set of all atomic literal sets of the feature model M and $\Pi_1(a)$ is a selection function that deterministically selects an element from a set a. The two sets of mandatory and dead features form separate atomic feature sets.

Excluding Parent-Child Interactions (PCI): this filter removes all parent-child interactions. Consequently, interactions such as the one between *Deployment* and *Web Application* in Figure 2.1 are excluded. We use the following definition for the corresponding filter ∇_{PCI}:

$$\nabla_{PCI}(I) = \{i \in I \mid \forall pci \in l_{parentChild(M)} : pci \not\subseteq i\}$$

where $l_{parentChild(M)}$ is the set of all parent-child literal sets of the feature model M.

Throughout the thesis, we use metrics that are a combination of the aforementioned filters and can be seen in Table 2.1.

3

Methodology

This chapter describes the methodology used to evaluate the coverage properties of sampling strategies for configurable software systems. Our evaluation focuses on understanding how different coverage metrics assess sampling strategies.

3.1 Research Questions

Sampling strategies are widely used to minimise the cost of analysing configurable systems by selecting a limited subset of valid configurations for measurement [7, 37, 46, 47]. Each sampling strategy exhibits specific strengths and weaknesses with respect to interaction coverage, configuration diversity, and computational effort. Consequently, selecting an appropriate sampling strategy poses a central challenge in the analysis of highly configurable systems.
Previous work has investigated sampling strategies using different evaluation criteria, such as prediction accuracy, robustness, and configuration representativeness [37, 52]. However, there is no consistency on how to precisely quantify partial t-wise interaction coverage with respect to an underlying feature model. This lack of agreement motivated Böhm et al. [15] to propose alternative coverage metrics for feature interactions, which are significantly different in their definitions and underlying assumptions.
We design our evaluation to systematically analyse how different coverage metrics influence the reported interaction coverage results of sampling strategies. Specifically, we apply three commonly used sampling strategies, namely solver-based, distance-based, and random sampling (see Section 2.2), to multiple real-world configurable software systems of varying configuration options and valid configurations (see Table 3.1). For each sampling strategy and system, we generate samples of valid configurations and compute their interaction coverage using some of the coverage metrics proposed by Böhm et al. (see Section 2.3).

Correlation between coverage metrics. By comparing the resulting coverage values, we analyse the relationship between the coverage metrics in terms of coverage computed for the same sampling strategy. In particular, we investigate whether different metrics yield consistent or divergent coverage results and whether these correlations remain stable across different sampling strategies. Strong correlations between metrics may indicate redundancy in the reported results, suggesting that simpler or more efficient metrics could serve as substitutes for more complex ones without a loss of explanatory power. By contrast, weak correlations or conflicting results suggest that the metrics capture fundamentally different aspects of interaction coverage.

RQ1 (Coverage Metric Relationships): Are there correlations between coverage metrics that are consistent across different sampling strategies?

Correlation between sampling strategies and coverage. Another crucial aspect is to understand whether there are correlations between sampling strategies and the coverage results determined by various metrics. If such a correlation exists, we can interpret the coverage analysis more efficiently and with greater confidence. To uncover these relationships, we aim to investigate whether certain sampling strategies consistently achieve high coverage values across different metrics, or whether these values depend heavily on the chosen metric. By analysing these patterns, we aim to identify recurring relationships between sampling strategies and the behaviour of the metrics that influence the reported evaluation results. This leads to our second research question.

RQ2 (Metric–Strategy Result Relationships): Do certain sampling strategies achieve consistently high coverage values across different coverage metrics?

Impact of the coverage metric on the constraint-coverage relationship. Finally, another crucial aspect is how strongly the configuration space is restricted and whether this affects the coverage of the interactions. Sampling strategies are known to behave differently in highly constrained versus weakly constrained configuration spaces [33, 46], which may also influence the coverage results reported by different metrics. Using the data obtained from the previous analyses, we examine the correlation of a system's constraint strength and its achieved coverage, and whether the choice of coverage metric alters this relationship. This analysis forms the basis of our third and last research question.

RQ3 (Constraint Strength and Coverage): Does the choice of coverage metric affect the correlation of a system's constraint strength and its achieved coverage?

3.2 Operationalisation

To address our research questions, we design an experimental setup that systematically applies multiple coverage metrics to samples generated by different sampling strategies. Our goal is not to influence how sampling strategies select configurations, but to analyse how different coverage metrics interpret the resulting samples. The sampling strategies we use are random, solver-based, and distance-based sampling. For each sampling strategy, we compute interaction coverage using all coverage metrics described in Table 2.1, following the formal definitions introduced by Böhm et al. [15]. In addition, t-wise sampling is used as a reference. Because the algorithm for t-wise sampling is the only deterministic one, it is difficult to contrast the results with randomised algorithms. Coverage is computed independently for each metric on the same sample set, ensuring that differences in coverage values are solely due to the metrics themselves.

Sampling budget and comparability. To ensure a fair comparison between the sampling strategies, we determine the number of configurations per strategy. We use t-wise sampling

with $t \in \{1, 2, 3\}$ as a reference for determining the sample sizes. For each system and each value of t, we count the number of configurations $|S_t|$ needed to achieve t-wise coverage. We then let each of the other sampling strategies (random, solver-based, and distance-based sampling) generate its own set of exactly $|S_t|$ valid configurations. Afterwards, we compute the t-wise coverage to evaluate the quality of the generated samples. For both cases, we consider $t \in \{1, 2, 3\}$. To distinguish these two uses of t, we use t_s to denote the sample size, which fixes the absolute sample size $|S_t|$, and t_c to denote the coverage strength at which we evaluate the samples.

Robustness. As the implementation of random, distance-based, and solver-based sampling involves randomness, the coverage results can differ from one run to the next. To ensure the reliability of our results, we run the sampling process 100 times for each strategy, system, and absolute sample size $|S_t|$. Once all 100 samples have been generated for each strategy, we compute the interaction coverage metrics. To obtain a single representative value for each strategy-metric pair, we use the arithmetic mean for **RQ1** and **RQ3** to compute the mean of the coverage values across all 100 repetitions.

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Subject systems and execution. We use this procedure on the real-world configurable software systems listed in Table 3.1. The selected systems cover a wide range of constraint strengths, reflecting the extent to which dependencies and exclusions restrict the possible combinations of features. By including systems with weak, moderate, and strong constraints, we make sure we have a diverse and representative selection of evaluation scenarios. For each system, each sampling strategy, and each value of t, we compute the interaction coverage using all eight metrics. This results in a comprehensive dataset of coverage values, which allows us to compare how the metrics evaluate the same samples across different strategies and systems.

Table 3.1: Overview of the systems under consideration, with the number of valid configurations ($|C_M|$) and the number of configuration options ($|F|$)

System	$\lvert C_M \rvert$	$\lvert F \rvert$
7z	68 640	44
AJStats	65 536	20
BerkeleyDBC	2 560	18
clasp	3 368	33
Curl	768	13
Dune	2 304	32
Hipacc	13 485	54
HSMGP	3 456	33
HSQLDB	864	18
HyTeG	32	8
JavaGC	193 536	39
LLVM	1 024	11
lrzip	432	19
PKJab	72	11
Polly	60 000	40
SQLite	3 932 160	39
TriMesh	239 360	68
VP9	216 000	42
WGet	5 120	16
x264	1 152	16
z3	2 048	15

Analysis of metric relationships (RQ1). For our first analysis, we will focus on the relationships between coverage metrics. For each sampling strategy, we analyse how coverage values produced by different metrics relate to each other. In particular, we analyse correlations between metrics to determine whether certain metrics consistently produce similar results. Strong correlations indicate potential redundancy and suggest that simpler or more efficient metrics could replace more complex ones without any significant loss of information. Furthermore, we analyse whether these relationships remain consistent across different sampling strategies and sample sizes t_s, addressing **RQ1**. To achieve this, we rely on *Spearman's rank correlation coefficient* [61]. We use Spearman's rank correlation coefficient given the non-linear nature of software coverage data, which often exhibits ceiling effects and diminishing returns [20, 42]. As the absolute sample size increases, the marginal increase in coverage decreases and eventually stagnates near the theoretical limit of 100%. This results in a monotonic but non-linear distribution [20]. Unlike linear measures of correlation, Spearman's ρ assesses the strength of monotonic relationships using data

ranks, making it better suited to variables that do not follow a normal distribution or a constant rate of change [69]. This enables us to identify redundant metrics where a high correlation suggests that a more efficient metric could replace a complex one, and ensures that our results remain robust across different sampling strategies.

Metric interpretation across sampling strategies (RQ2). To this end, we examine whether coverage metrics consistently evaluate the three sampling strategies across different systems and sample sizes t_s. Specifically, we examine whether certain sampling strategies consistently achieve high or low coverage values across all metrics, or whether coverage assessments vary significantly depending on the chosen metric. By analysing these patterns, we aim to identify recurring relationships between sampling strategies and the behaviour of the metrics that influence the reported coverage results. Unlike **RQ1**, where we analyse system-level relationships based on aggregated coverage values, here we use all coverage values obtained from independent sampling repetitions. This can help us capture the variability inherent in stochastic sampling strategies and to compare their coverage behaviour at the distribution level. To address **RQ2**, we compare the coverage distributions of the different sampling strategies using the *Kruskal–Wallis test* [43, 51]. This non-parametric approach is selected because interaction coverage data often violates the assumption of normality due to ceiling effects and system constraints [20]. Following a significant Kruskal–Wallis result, we apply *Dunn's test* [24] for post-hoc pairwise comparisons to identify which strategies in particular differ. One potential error that can occur is a *Type I* error, which arises when a null hypothesis that is in fact true in the population is incorrectly rejected [8, 51]. To reduce the family-wise error rate and minimise the risk of Type I errors in multiple comparisons, we apply the *Holm–Bonferroni* correction [34]. This statistical approach allows us to account for the fluctuations associated with stochastic samples and to determine whether certain strategies consistently outperform others across different systems.

Impact of the coverage metric on the constraint-coverage relationship (RQ3). Finally, we investigate whether the choice of coverage metric influences the extent to which a system's configuration space is constrained and the coverage that is achieved. Rather than counting the constraints directly, we characterise the strength of a system's constraints in terms of the reduction in its configuration space. For each system i, we compute the ratio

$$R_i = \frac{|C_{M_i}|}{2^{|F_i|-1}}$$

where $|C_{M_i}|$ denotes the number of valid configurations for the i-th model and $|F_i|$ denotes the number of features for the i-th model. This ratio indicates the extent to which constraints limit the theoretically possible configuration space. Values close to one correspond to weakly constrained systems, while values close to zero correspond to highly constrained systems. Because R_i ranges over several orders of magnitude across our subject systems, we analyse it on a logarithmic scale ($\log_{10}(R_i)$).
To examine this, we treat $\log_{10}(R_i)$ as a continuous measure of constraint strength rather than discretising it into categories, thereby avoiding the loss of information and the arbitrary boundary placement that binning would introduce. For each coverage metric and each combination of coverage strength t_c and sample size t_s, we compute one coverage value per system by averaging over the sampling repetitions and strategies, and then quantify

the relationship between this system-level coverage and $\log_{10}(R_i)$ using Spearman's rank correlation coefficient ρ. We use Spearman's ρ for the same reasons as in **RQ1** (see paragraph 3.2): it captures monotonic relationships without assuming linearity or normality, both of which are violated by interaction coverage due to ceiling effects. A correlation is considered statistically significant at $p < 0.05$. This allows us to assess, separately for each metric, whether more strongly constrained systems systematically achieve higher or lower coverage, and whether this sensitivity to constraint strength differs across metrics.
Overall, this operationalisation allows for a controlled and systematic comparison of coverage metrics, enabling us to evaluate their relationships, redundancies, and applicability across different sampling strategies and system characteristics.

4 Evaluation

4.1 Results

This section presents the results of our evaluation of coverage metrics and sampling strategies. We first examine the relationships between the coverage metrics and analyse how consistently they agree across different experimental settings. We then investigate how the choice of metric influences the assessment of sampling strategies and whether certain strategies achieve consistently higher coverage than others. Finally, we analyse whether constraint strength is associated with systematic differences in achieved coverage.

Missing Data. To determine the sample size, we generate samples using t-wise sampling. For two systems, TriMesh and SQLite, this sampling exceeded our timeout limit of 24 hours at the highest sample size $t_s = 3$, as the number of t-wise interactions to be covered grows too large for these systems at that strength. Consequently, these two systems are excluded from all evaluation conditions with $t_s = 3$, which therefore comprise 19 rather than 21 systems. All conditions with $t_s = 1$ and $t_s = 2$ include the full set of 21 systems. We point out explicitly wherever this reduced set affects the interpretation of the results. This is particularly relevant for **RQ3**, as both excluded systems are among the most heavily constrained.

4.1.1 RQ1: Coverage Metric Relationships

To address our first research question, we analyse the relationships between the coverage metrics introduced in Section 2.3. We therefore compare all metric pairs using Spearman's rank correlation coefficient. The resulting coefficient ρ indicates how strongly two metrics are monotonically related.
Figure 4.1 shows the mean Spearman correlation of each metric pair, averaged across the three sampling strategies, for each combination of coverage strength t_c and sample size t_s. Overall, all metric pairs show a positive correlation, and across all 756 evaluated settings (28 metric pairs over 27 combinations of strategy, t_s, and t_c) every correlation is statistically significant ($p < 0.05$). Although this involves a large number of comparisons, we do not apply a correction for multiple tests here: the metrics are derived from overlapping interaction filters and are therefore inherently correlated with one another, meaning that the significance of each individual correlation is not a random finding that could be inflated by the number of tests. The highest average correlation can be observed between MDAP

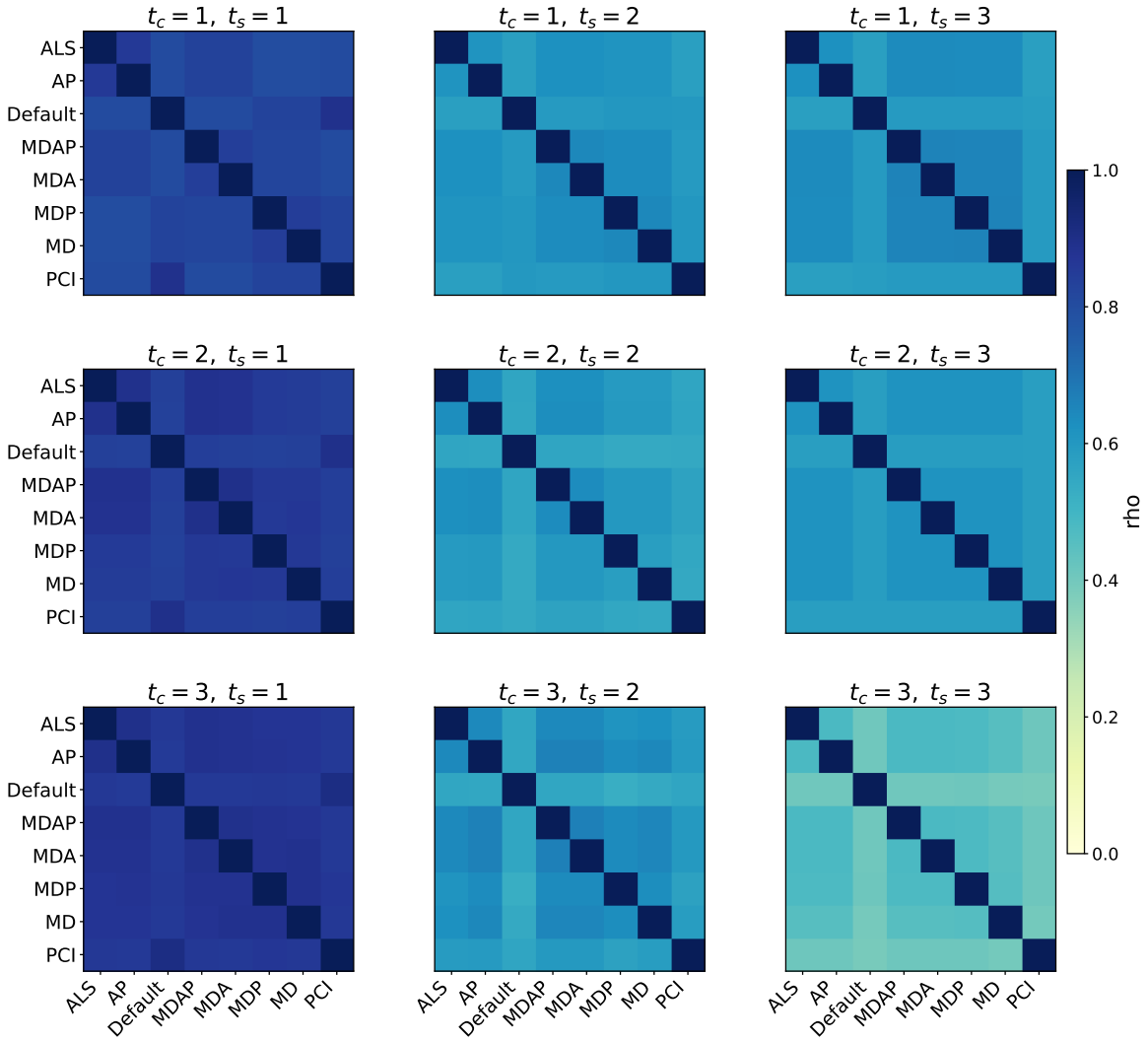
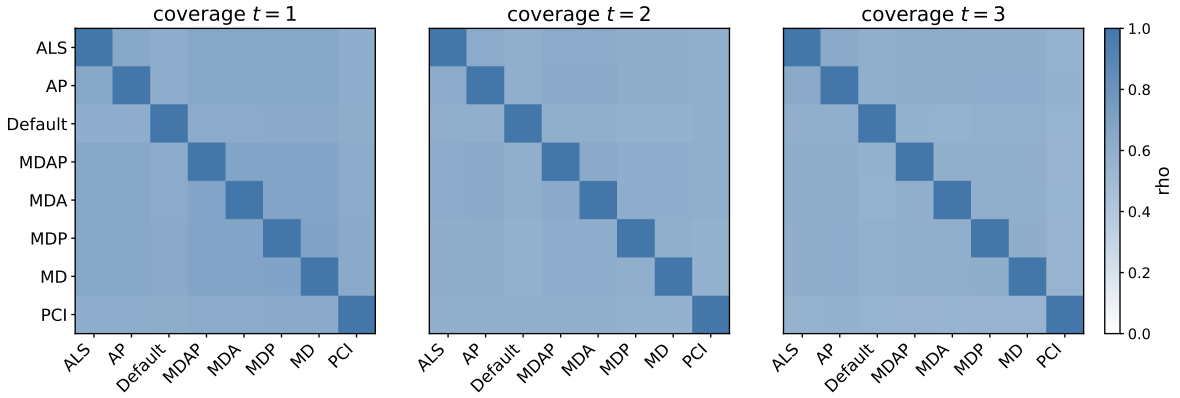


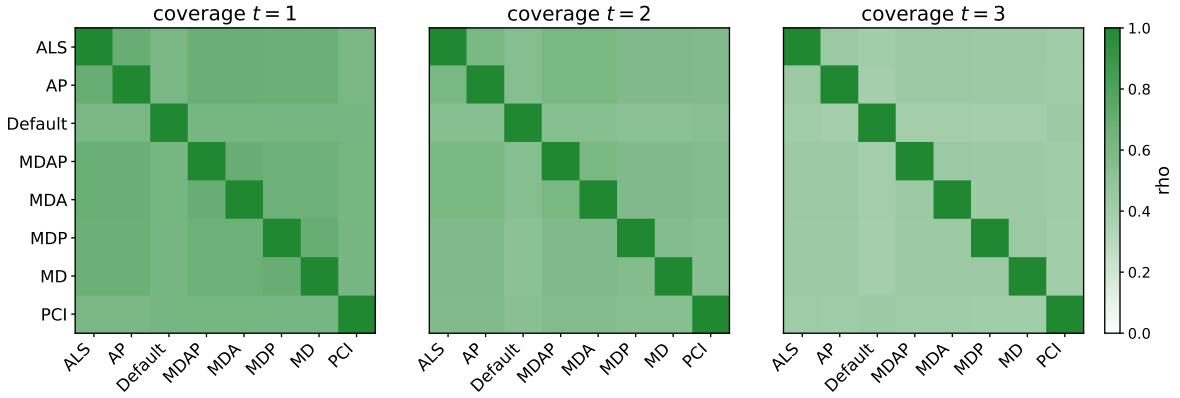
Figure 4.1: Mean Spearman correlation between all pairs of coverage metrics, averaged across the three sampling strategies, for each combination of coverage strength t_c and sample size t_s.

and MDA at $\rho = 0.722$, whereas the correlations between DEFAULT and the remaining metrics are only slightly lower, ranging from $\rho = 0.678$ to $\rho = 0.698$. Among these, the strongest relationship can be observed between DEFAULT and PCI. Overall, the correlations lie roughly between $\rho = 0.678$ and $\rho = 0.722$, and no pair reaches strong agreement on average. This suggests that, while the metrics are clearly interrelated, this relationship is not strong enough to regard them as interchangeable.

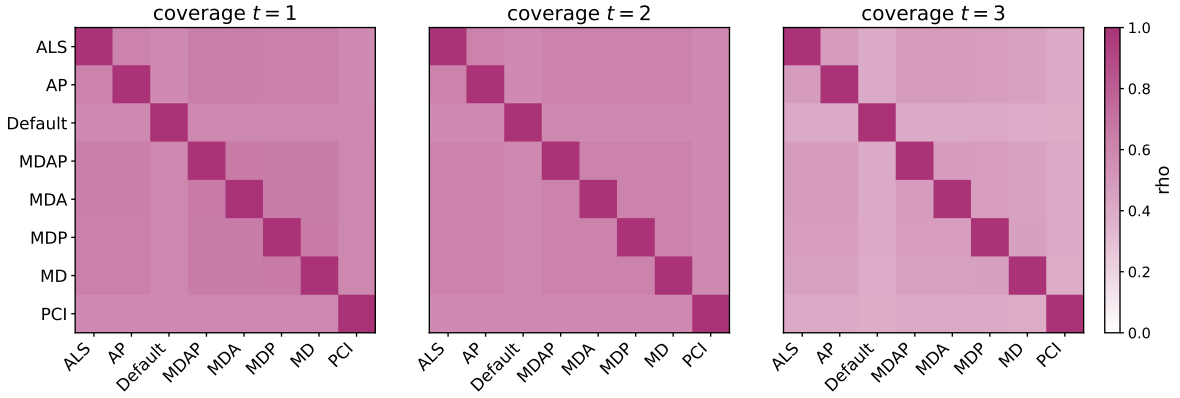
To analyse the dependence of these relationships on the experimental conditions in more detail, we examine how much the agreement varies across the individual conditions. Although the average correlations in Figure 4.1 are consistently high, this average value hides how much the degree of agreement varies across the individual conditions. To make this variation not only visually apparent but also measurable, we classify each of the 756 correlations based on the absolute value of their Spearman's coefficient according to standard



(a) Random sampling, $t_s = 3$.



(b) Distance-based sampling, $t_s = 3$.



(c) Solver-based sampling, $t_s = 3$.

Figure 4.2: Metric agreement heatmaps for the three sampling strategies for sample size $t_s = 3$.

thresholds: strong ($|\rho| \geq 0.80$), moderate ($0.60 \leq |\rho| < 0.80$) and weak ($|\rho| < 0.60$). In total, 241 are strong, 315 are moderate and 200 are weak or very weak, which shows that the correlation is generally moderate and that the strong correlations are concentrated in the simplest conditions.

The degree of metric agreement also depends on the sampling strategy from which the samples are drawn. Figure 4.2 shows one heatmap per sampling strategy, with the sample size fixed at $t_s = 3$ and the three coverage strengths $t_c = 1, 2, 3$ shown side by side. The corresponding heatmaps for $t_s = 1$ and $t_s = 2$ are provided in Section A.1.2. The three strategies differ significantly in terms of how stable this agreement is. Random sampling is the most stable: its metric agreement remains comparatively consistent across all conditions (mean $\rho = 0.746$) and weakens only gradually as the sample size t_s and coverage t_c increase. Distance-based sampling is the most sensitive: although its correlation is highest at the simplest setting, it declines most rapidly, so that its mean correlation coefficient at the largest sample size t_s falls to $\rho = 0.551$ and thus falls below the threshold for moderate, even though its overall mean ($\rho = 0.685$) remains slightly above that of solver-based sampling. Solver-based sampling lies between these two extremes in terms of the rate at which its agreement declines, but exhibits the lowest overall correlation of the three strategies (mean $\rho = 0.667$), meaning that, on average, its metrics agree the least, even though its decline is not the steepest. The detailed values for all three strategies per condition are listed in Section A.1.2.

A recurring pattern involves the relationship between Default and PCI, which remains relatively strong even when the correlation between other pairs of metrics weakens. This follows from the definition of PCI: since it only filters interactions of size at least two, Default and PCI are identical for $t_c = 1$ and diverge only at higher coverage strengths. Consequently, under the simpler conditions, the two metrics follow each other more closely than most other pairs, and their agreement gradually decreases as t_c increases.

Overall, the heatmaps show that the coverage metrics correlate positively with one another across all sampling strategies and all sample size conditions. However, the strength of this correlation is not constant. Random sampling appears to be the most consistent, distance-based sampling shows the greatest variation in sample sizes and coverage strengths, and solver-based sampling lies between these two extremes. As already indicated by Figure 4.1 the weakest average correlations occur between the Default metric and the remaining coverage metrics. To examine these weaker relationships in more detail, Figure 4.3 compares the signed differences between Default and each of the remaining metrics across the sampling strategies, sample sizes, and coverage strengths.

Across most of these box plots, the signed differences remain closer to zero when coverage is evaluated at $t_c = 1$. As the evaluation strength increases to $t_c = 2$ and $t_c = 3$, the differences usually become more spread out, and unusually large deviations occur more often. In addition, the median differences often shift in similar directions across neighbouring conditions, which suggests that these deviations are systematic rather than random.

A particularly notable exception is PCI in Figure 4.3. For $t_c = 1$, the signed differences are exactly zero. This can be explained by the definition of PCI. Since PCI filters out interactions that contain both a parent and its child, it only affects interactions of size at least 2. Consequently, Default and PCI produce identical values for $t_c = 1$, while visible and often negative differences emerge for $t_c = 2$ and $t_c = 3$.

This behaviour differs from the remaining metrics, such as ALS in Figure 4.3. Although Default and ALS often produce similar coverage values, these values are not exactly equal: for the same sample, the two metrics still assign slightly different coverage numbers. Here as well, the spread of the differences increases for larger sample sizes and larger coverage

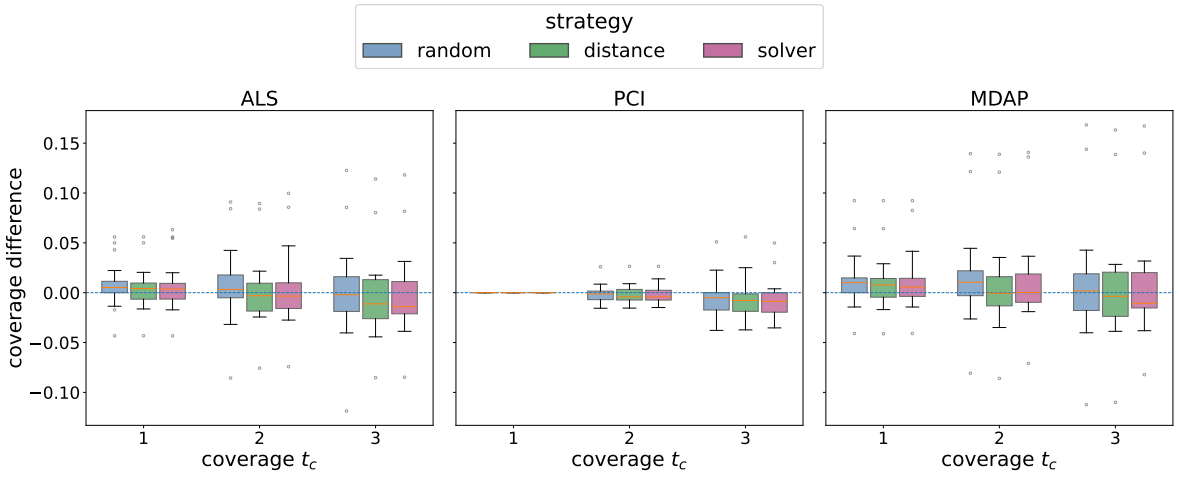


Figure 4.3: Signed coverage differences between DEFAULT and ALS, PCI, and MDAP for $t_s = 3$.

strengths. In addition, the differences produced by random sampling tend to remain closer to zero, whereas distance-based and solver-based sampling tend to show stronger deviations for larger sample sizes and higher coverage strengths.
The effect becomes even more visible for metrics that combine several filters. For example, MDAP in Figure 4.3 shows that the signed coverage differences between DEFAULT and MDAP becomes larger and that the positive outliers become considerably more pronounced. Thus, while the correlation analysis shows that the metrics tend to move together, the signed-difference box plots reveal that they can still differ systematically in magnitude and direction.
Overall, the results for **RQ1** show that the coverage metrics are consistently positively related, but their agreement is only moderate, reaching strong agreement only in the simplest conditions, and depends on both the sampling strategy and the two experimental dimensions, namely the sample size and the evaluation strength. We further demonstrate that these metrics are not numerically interchangeable, especially for larger sample sizes and higher coverage strengths.

4.1.2 RQ2: Metric–Strategy Result Relationships

To answer our second research question, we analyse how the ranking of random, distance-based, and solver-based sampling changes across the coverage metrics for different values of t_s and t_c.
Across all metrics, a common overall pattern can be observed: random sampling usually achieves the highest median coverage values, distance-based sampling usually ranks second, and solver-based sampling tends to perform worst. At the same time, the absolute coverage values depend much more strongly on t_s and t_c than on the chosen metric.
This general order holds true for all three sample sizes t_s. The median coverage values (Figure 4.4) show that random sampling generally has the highest median coverage values, distance-based sampling is below this, and solver-based sampling usually has the lowest values. Increasing the sample size t_s raises the overall coverage level, but leaves this order

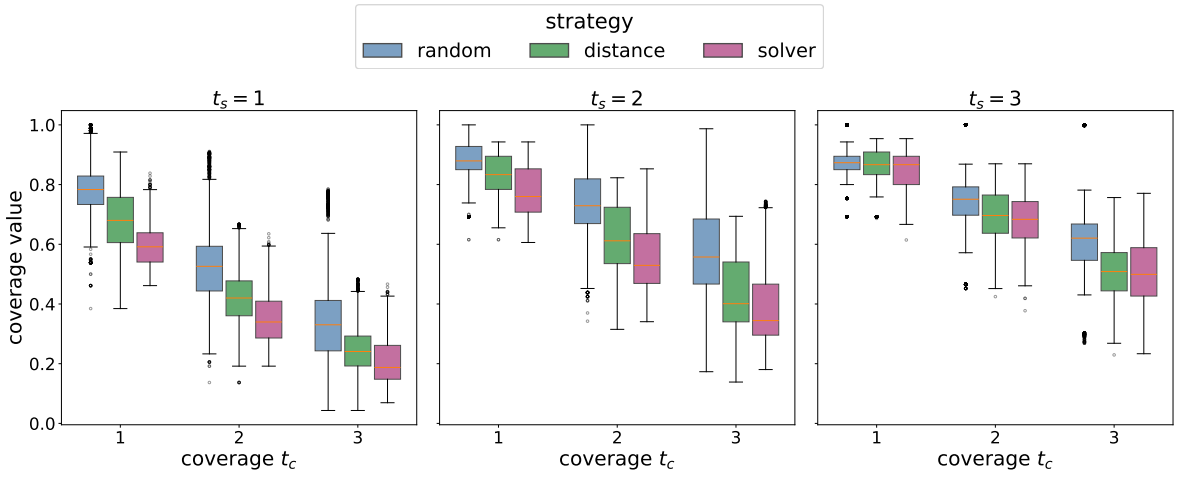


Figure 4.4: Coverage distributions of the DEFAULT metric across sampling strategies, per sample size t_s.

unchanged. What does change with the coverage level is the degree to which the strategies stand out from one another: at $t_c = 1$, the three strategies are closely clustered, while at $t_c = 3$ the gaps widen and solver-based sampling consistently achieves the lowest median coverage values, while random sampling continues to perform best. The ranking also remains stable across different metrics: although individual ranking positions occasionally change, random sampling occupies many of the top positions, solver-based sampling is concentrated in the lower part of the ranking, and distance-based sampling lies in between. The differences between the metrics are therefore minor and of a systematic rather than arbitrary nature.

Beyond the median ordering, the same box plots also reveal how the full coverage distributions behave. For lower values of t_c, the box plots of the three strategies often overlap more strongly, which indicates that their coverage distributions are closer to one another. For higher values of t_c, the differences between the strategies become more visible, especially because solver-based sampling shows a stronger decrease in median coverage values than random and distance-based sampling as t_c increases. Random sampling generally shows the highest median values for coverage with comparatively low variation, which means that its coverage values vary only slightly between repetitions, while distance-based sampling is often closer to random sampling than to solver-based sampling. The Kruskal–Wallis test confirms these observations. Across all 72 combinations of metric, sample size, and coverage strength, the differences between the three strategies are statistically significant ($p < 0.05$ in every case), with effect sizes ranging from negligible ($\epsilon^2 \approx 0.01$) to large ($\epsilon^2 \approx 0.438$, mean ≈ 0.19). Dunn's post-hoc test with Holm correction supports the expected ordering: random sampling attains the highest median coverage in all 72 settings, and the full ordering random sampling $>$ distance-based sampling $>$ solver-based sampling is statistically significant (all pairwise Dunn comparisons with $p < 0.05$) in 56 of them. This number is identical for every one of the eight metrics (7 of 9 settings each), and the two exceptions per metric always occur in the same sample size $t_s = 3$, where the strategies converge and where two systems are additionally excluded for performance reasons (see Section 4.3.1): at $t_c = 1$ the medians of distance-based and solver-based sampling become tied or invert, so although

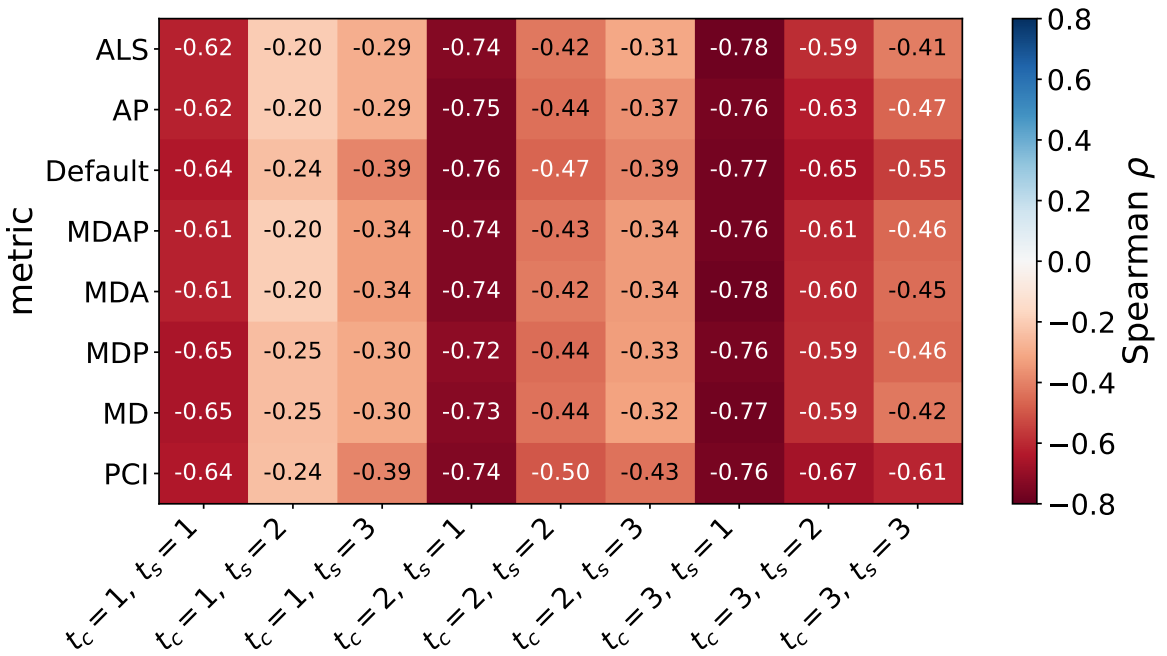


Figure 4.5: Spearman correlation ρ between system-level coverage and R_i for each coverage metric and each combination of coverage strength t_c and sample size t_s. All correlations are negative.

random sampling remains significantly highest, the strict three-way ordering no longer holds, while at $t_c = 3$ distance-based and solver-based sampling are no longer significantly separated from each other. The choice of metric therefore leaves the comparative ranking essentially unchanged.

Overall, the results for **RQ2** show that changing the coverage metric usually does not change the overall ranking of the sampling strategies. Across almost all sample sizes and coverage strengths, random sampling achieves the highest coverage values, distance-based sampling usually ranks second, and solver-based sampling tends to perform worst. At the same time, the results also show that the sample size and coverage strength have a stronger influence on the absolute coverage values than the specific coverage metric.

4.1.3 RQ3: Constraint Strength and Coverage

To address our third and final research question, we analyse whether the coverage values of a system are dependent on the constraint strength of the system. We use R_i to measure the constraint strength of a system. To measure the relationship between system-level coverage and constraint strength, we use Spearman's rank correlation coefficient ρ for each metric separately and each combination of coverage strength t_c and sample size t_s. Smaller values of R_i indicate that the system is more heavily constrained. Thus, a negative ρ means that more constrained systems achieve higher coverage.

Figure 4.5 shows the resulting correlation for every metric, sample size t_s, and coverage strength t_c. Across all 72 evaluated settings (8 metrics over 9 combinations of t_c and t_s), the

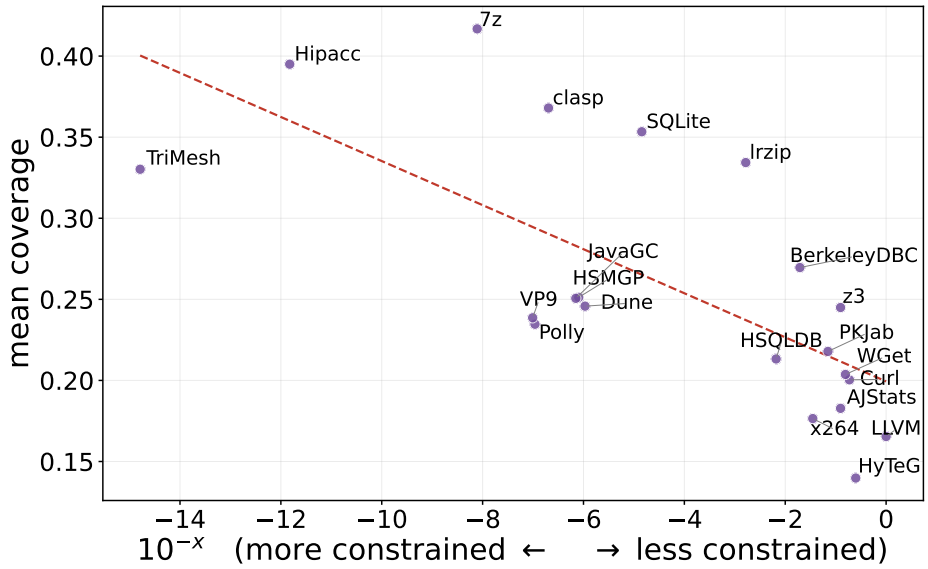


Figure 4.6: System-level Default coverage against constraint strength R_i at $t_c = 3$, $t_s = 1$. Smaller R_i indicates a more heavily constrained system. The dashed line shows the linear trend; the strength of the association is given by Spearman's $\rho = -0.77$.

heatmap shows that every correlation is negative. As mentioned before, negative correlations indicate that more heavily constrained systems consistently achieve higher coverage. The relationship is moderate overall (mean $\rho = -0.509$), ranging from $\rho = -0.196$ to $\rho = -0.777$, and is statistically significant ($p < 0.05$) in 42 of the 72 settings. The most heavily constrained systems, TriMesh ($\log_{10}(R_i) = -14.8$) and Hipacc (-11.8), lie at the high-coverage end, whereas LLVM (0.0), whose configuration space is essentially unrestricted, lies among the lowest.

In order to analyse the relationship between system-level coverage and constraint strength across the experimental conditions, we compare the correlations separately for each sample size. It becomes obvious that the strength of the correlation is heavily dependent on t_s. When the sample size is fixed at $t_s = 1$, the correlation becomes strong and uniformly significant across all metrics and coverage strengths (mean $\rho = -0.712$, all 24 correlations significant). For $t_s = 2$, the agreement weakens and becomes mixed (mean $\rho = -0.427$, 13 of 24 significant), and for $t_s = 3$ it weakens further (mean $\rho = -0.387$, only 5 of 24 significant). As mentioned before, the systems TriMesh and SQLite exceeded the sampling timeout at $t_s = 3$, which is why they are not included in these settings ($n = 19$). As both lie at the constrained end of the axis, their absence partly accounts for the weaker correlation under $t_s = 3$.

Within each t_s, the correlation increases as t_c increases. For $t_s = 1$, the mean correlation increases from $\rho = -0.631$ at $t_c = 1$ to $\rho = -0.738$ at $t_c = 2$ and $\rho = -0.767$ at $t_c = 3$. The two dimensions thus strengthen each other: the relationship between constraint strength and coverage is strongest where the sample size is smallest and the coverage strength is highest.

However, comparing the results from all eight metrics with each other, it becomes clear that the choice of metrics has little effect on this relationship. Across all nine settings, the eight

metrics show correlations that fall within a close range, and their mean correlations cluster between $\rho = -0.484$ (ALS) and $\rho = -0.553$ (PCI). The agreement between metrics is closest precisely where the relationship is strongest: at $t_c = 3$, $t_s = 1$, the eight metrics span only 0.017 (ρ between -0.760 and -0.777), and the three $t_s = 1$ settings together never exceed a spread of 0.042. This strongest setting ($t_c = 3, t_s = 1$) is shown in Figure 4.6. The metrics diverge only in the weaker settings: the largest spreads (0.098, 0.125, and 0.200) all occur at $t_s = 3$, where the correlations are themselves small and therefore more sensitive to the placement of individual systems.
Overall, the results for **RQ3** show that more heavily constrained systems consistently achieve higher coverage, that this relationship strengthens as the coverage strength increases and as the sample size decreases. However, the relationship is largely independent of the choice of coverage metric. This is consistent with the moderate agreement between the metrics observed in **RQ1**: although the metrics are not numerically interchangeable, they respond to constraint strength in essentially the same way.

4.2 Discussion

In this section, we interpret the results of our analysis in the context of the research questions. We discuss what the observed relationships between the coverage metrics mean for their practical application, how the choice of the included metric affects the evaluation of sampling strategies, and to what extent the strength of the constraints influences the generalisability of our results.

4.2.1 RQ1: Coverage Metric Relationships

The results of **RQ1** show that the different coverage metrics are clearly related, since all metric pairs exhibit positive correlations. However, these correlations range from moderate to strong and are not nearly perfect. This suggests that while the metrics capture overlapping aspects of sample quality, they do not represent the same concept in an interchangeable way. In other words, the results suggest that coverage is not a neutral or metric-independent property of a sample. Instead, the reported coverage depends on how the metric is defined, which interactions are included and excluded.
This effect increases with the complexity of the evaluation scenario: as sample size t_s and coverage t_c increase, the metrics become less consistent, and their numerical differences grow. The reason for this is likely systemic: in simple interaction spaces, most metrics measure similar things, but as the space grows in size, the different assumptions behind the individual filters become more significant, meaning that the reported coverage depends more heavily on the chosen metric. This effect also depends on the sampling strategy: it is smallest for random sampling and largest for distance-based sampling, suggesting that a metric's influence is shaped not only by the metric itself but also by the structural properties of the samples generated by a given strategy. It is important to note that agreement in correlation and agreement in absolute value are not the same: two metrics can rank systems almost identically yet assign systematically different coverage values, so that agreement in

trend does not imply that the strength of a sample's evaluation also agrees.
The PCI metric is the clearest example that these differences are not arbitrary, but meaningful. By definition, it coincides with DEFAULT when $t_c = 1$, as in this case there are no higher-order interactions that need to be filtered out, and it only diverges for larger interaction sizes, at which point its filter begins to exclude interactions that DEFAULT continues to take into account. Its behaviour corresponds exactly to this definition: identical to DEFAULT when $t_c = 1$ and systematically different beyond that. The differences between the metrics therefore reflect the intended meaning of the corresponding filters. The same general conclusion applies to the other metrics, that contain filters, as well: metrics that remain relatively close to the DEFAULT, such as ALS, still lead to systematic differences, while metrics that combine multiple filters differ more significantly. Thus, the more selectively a metric defines the relevant interaction space, the more it can alter the reported coverage values.

RQ1 (Coverage Metric Relationships): The coverage metrics show a consistent positive correlation, demonstrating that they capture related aspects of sample quality. However, they are not interchangeable, as both their agreement in ranking and their numerical agreement vary depending on the evaluation conditions and sampling strategies. Thus, the evaluation of sampling strategies is not metric-neutral: different coverage metrics can lead to significantly different evaluations of the same sampler, especially under more complex evaluation conditions.

4.2.2 RQ2: Metric–Strategy Result Relationships

The results of **RQ2** suggest that the comparison of sampling strategies with regard to the chosen coverage metric is largely robust. For almost all metrics, the same qualitative ranking of strategies can be observed: random sampling achieves the highest median coverage values in most cases, distance-based sampling generally ranks second, and solver-based sampling typically achieves the lowest median coverage values. The statistical tests confirm this consistency, as the expected order random sampling $>$ distance-based sampling $>$ solver-based sampling occurred in 7 out of 9 settings for each of the eight metrics. With the exceptions being identical in every case, the ranking patterns therefore remain almost identical across the metrics and are not merely visually similar. A plausible explanation for this order lies in how widely each strategy spreads its configurations across the valid configuration space. As explained in Section 2.2, solver-based sampling tends to generate structurally similar configurations clustered together, while distance-based sampling purposefully designs the sample to follow a distance distribution rather than to cover interactions. Random sampling, on the other hand, is distributed across all valid configurations without such bias, so that, for a sample of the same size, it tends to cover a greater range of feature interactions. It is therefore not surprising that random sampling achieves the highest coverage of interactions, even though it is the simplest of the three strategies. It should be noted that none of the three strategies is designed to maximise interaction coverage (see Section 4.3.2). The ranking therefore reflects the relative coverage behaviour of general-purpose samplers rather than performance against a coverage-optimised baseline.

At the same time, the absolute coverage values depend far more on the evaluation scenario than on the metric: the strategies are almost identical at $t_c = 1$ and only begin to diverge significantly at $t_c = 3$, where the weaker performance of solver-based sampling becomes apparent. Demanding coverage requirements are therefore more useful for distinguishing between strategies than simple ones. The metric, on the other hand, shifts the individual rankings only slightly and rarely changes which strategy is evaluated as the best.
If we compare these findings with Section 4.2.1, an interesting correlation emerges. In **RQ1**, random sampling showed the most consistent pattern of agreement across all coverage metrics, which is consistent with the fact that it also proves to be the most stable strategy in **RQ2**. Distance-based sampling, in contrast, was the most sensitive strategy with respect to metric agreement in **RQ1**. In **RQ2**, however, distance-based sampling still performs between random and solver-based sampling and overall remains closer to random sampling. However, solver-based sampling occupies the middle position in terms of metric sensitivity, between random and distance-based sampling, yet still performs worst overall in terms of coverage. The sensitivity ordering from **RQ1** and the performance ordering from **RQ2** do not coincide: the most metric-sensitive strategy, distance-based sampling, is only the middle performer, while the less sensitive solver-based sampling performs worst. This shows that metric sensitivity and absolute coverage performance are related but distinct aspects of sampling-strategy behaviour.

RQ2 (Metric–Strategy Result Relationships): Random sampling most consistently achieves the highest coverage values, while distance-based sampling usually ranks second and solver-based sampling tends to perform worst. Although certain scenarios, particularly a low coverage strength combined with a large sample size, can reduce these differences, the overall ordering remains very similar across all eight coverage metrics for the three sampling strategies studied here. The choice of metric therefore rarely changes the ranking of the strategies, even though the metrics are not numerically interchangeable: what determines how clearly the strategies can be distinguished is the difficulty of the evaluation scenario, especially a high coverage strength.

4.2.3 RQ3: Constraint Strength and Coverage

The results of **RQ3** show that the choice of coverage metric has little influence on the relationship between the constraint strength and the achieved coverage. The relationship between constraints and coverage itself is clear and consistently negative. What remains essentially unchanged is the way in which this relationship is represented across the eight metrics. The slight differences that remain between the metrics are due to rank sensitivity: since the metrics assign slightly different absolute coverage values to the same sample (Section 4.2.1), the ranking of the systems by coverage can vary slightly between the metrics. Such rank changes occur only when systems achieve closely spaced coverage values: where coverage is well separated, the ordering is robust regardless of metric, but where coverage values lie close together, small metric-induced differences are enough to swap ranks. This is why the largest differences between the metrics' correlation values occur at the highest sample size $t_s = 3$, where the correlations are weakest and the coverage values most tightly

bunched.

This behaviour follows from what the metrics measure: all eight are filters over the same set of t-wise interactions, differing in which interactions they count, not in the configuration space they evaluate. The relationship examined here is between coverage and the constraint strength of a system, which is a property of that configuration space and is independent of any metric. Consequently, while the metrics assign different absolute coverage values to the same sample, as shown in **RQ1**, they all respond to changes in constraint strength in the same direction and to a similar degree. The constraint-coverage relationship is therefore a property of the systems rather than of the chosen metric.

This is consistent with the findings of **RQ1** and **RQ2**. **RQ1** shows that the metrics are not numerically interchangeable, agreeing only moderately in their absolute values. **RQ2** shows that, despite this, the ranking of the sampling strategies is stable across metrics.

That the more strongly constrained systems achieve higher coverage values can be explained by the size of space that is covered. The constraints exclude a large portion of the feature combinations that are theoretically possible. As a result, the set of valid t-wise interactions that the metric is filtering out is smaller. This means that a sample covers a large portion of this reduced space, which has a higher coverage value. In contrast, weakly constrained systems have a much larger valid interaction set, which means that a sample of similar size covers a smaller portion of it. As such, the negative correlation between R_i and the coverage values is a result of how much the constraints shrink the space to be covered, rather than our selected sampling strategies or metrics. This also explains why the effect becomes much more noticeable with the coverage strength t_c: a higher value of t_c increases the interaction space, which makes the difference between strongly and weakly constrained systems much more pronounced.

For the practical evaluation of the quality of a sample, this means that the coverage values cannot be compared across systems without taking the constraint strength into account. High coverage values from a strongly constrained system and lower values from a weakly constrained system could be an indication of the size of their respective interaction space rather than a difference in sample quality. Since this effect is basically the same for our eight metrics, it cannot be avoided by simply choosing a different metric. As with the ranking of the sampling strategies, the choice of metric changes the absolute coverage values but not the underlying relationship. Taken together, the metrics differ in the absolute values they produce, but not in the conclusions they lead to.

RQ3 (Constraint Strength and Coverage): More heavily constrained systems consistently achieve higher coverage, and this relationship is essentially the same for all eight metrics. The choice of metric does not change it. What does change it is the evaluation scenario, the sample size t_s and the coverage strength t_c. The practical consequence is that coverage values cannot be compared across systems without accounting for their constraint strength: a higher value on a strongly constrained system need not indicate a better sample, but may simply reflect its smaller interaction space. Choosing a different metric does not remove this effect.

4.3 Threats to Validity

In this section, we discuss potential threats to the validity of our evaluation and analysis. We distinguish between threats to internal and external validity. Internal validity refers to the accuracy and reliability of our experimental design and measurement procedures. External validity refers to the generalisability of our results to other configurable systems and sampling strategies.

4.3.1 Internal Validity

The implementations of random, distance-based, and solver-based sampling strategies involve stochastic components, which can lead to variations in coverage results across different runs. To reduce the influence of random variations, we performed 100 independent repetitions for each strategy, system, and sample size. We then analysed the resulting coverage distributions using non-parametric statistical methods, which ensured robust and reliable conclusions.

To ensure comparability between the different sampling strategies, we determined sample sizes based on t-wise sampling with $t \in \{1, 2, 3\}$. For each system and each value of t, we measured the number of configurations $|S_t|$ needed to achieve t-wise coverage. All other sampling strategies generated exactly $|S_t|$ valid configurations. While larger sample sizes might reveal additional behavioural characteristics of certain strategies, fixing the absolute sample size allows for a controlled and fair comparison between the approaches.

As part of our experiment, we used a greedy implementation of t-wise sampling to determine the sample size budgets. For the TriMesh and SQLite systems, the time limit was exceeded when generating t-wise samples for $t_s = 3$, which is why these two systems were excluded from the $t_s = 3$ conditions. Since both systems are relatively large, their absence could, in principle, influence the results for this condition. However, the trends observed for all three research questions are consistent across all available systems, sampling strategies, and coverage levels, so we assume that they also apply to the two missing systems.

Our conclusions depend on the appropriate selection and application of statistical tests. Although we used non-parametric methods to account for non-normal coverage distributions and applied corrections for multiple comparisons, alternative statistical choices could influence the observed strength or significance of the correlations.

Finally, our evaluation is based on the correct implementation of Binary Decision Diagrams, the accurate computation of interaction coverage, the correct application of filters, and the correct generation of coverage metrics. Errors in the coding of feature model constraints, the enumeration of valid configurations, the generation of interaction sets, or the application of filters could distort the coverage values and consequently compromise the statistical analyses and the conclusions derived from them. To minimise this risk, we validated our implementation through integration and component tests, as well as by comparing selected results with previously published findings.

4.3.2 External Validity

In this thesis, we selected a subset of interaction filters identified by Böhm et al. [15] and derive coverage metrics based on these filters. Consequently, not all possible filter combinations are taken into account, which could limit the generalisability of our results to alternative coverage definitions, since a filter that we have not included may respond differently to the sampling strategies than the metrics we have considered. In particular, certain filters require additional domain knowledge or specific annotations of the feature model (e.g., explicitly marked abstract features) which are not available in our setting.
In our evaluation, we limited the selection of sampling strategies to random, distance-based, and solver-based sampling. This could affect the generalisability of our findings to sampling strategies not considered in this thesis. However, the sampling strategies we selected are the most common and cover a broad spectrum of techniques.
A related limitation concerns the interpretation of our **RQ2** robustness finding. The three strategies compared here differ substantially in their interaction coverage, and across our evaluation conditions this separation was large enough that a change of metric rarely reordered them. For a different or larger set of strategies, particularly strategies whose coverage profiles lie closer together, the metric choice could have a stronger influence on the ranking. Our results therefore show that the ranking of these three common strategies is stable across coverage metrics. Whether the same holds for strategies with more similar coverage profiles remains open.
Finally, we conducted our evaluation on 21 real-world configurable software systems. Although these systems vary in terms of size, number of features, and number of valid configurations, they cannot fully represent the diversity of all configurable systems. The selected systems could be biased towards specific domains, structural characteristics, or constraint strength. To minimise this risk, we deliberately included systems that cover a broad range of configuration space sizes and number of features.

4.4 Future Work

As discussed in Section 4.3, our study has some limitations, some of which suggest a direction for future work.
We compared only general-purpose sampling strategies, none of which are designed to maximise interaction coverage. An obvious extension would be to examine coverage-optimising sampling strategies in their own right and to compare several such strategies against one another using different coverage metrics. Since each coverage-optimised sampler is built to maximise a particular notion of interaction coverage, and thus implicitly a particular coverage metric, this would reveal whether the choice of metric influences which coverage-optimising strategy performs best, similar to the analysis we have presented here for general-purpose strategies.
Our distinction between strongly and weakly constrained systems is defined relative to the spectrum of constraint conditions observed in our 21 systems, and is therefore dependent on this specific sample. Although this range already extends from systems that are essentially

unconstrained to those that are strongly constrained, an extension to include systems with even greater constraints and a more diverse range of application areas would make it possible to test the negative correlation between the degree of constraint and coverage beyond the conditions discussed here, and to reduce the risk of bias in favour of the areas and structural characteristics represented in our current dataset.
Our metrics cover some of the interaction filters proposed by Böhm et al. [15], as the filters not included require annotations of the feature model, such as explicitly marked abstract features, which our models do not provide. Constructing or inferring these annotations for our 21 systems would make the remaining metrics computable and would show whether the agreement between metrics, and their limited influence on the strategy ranking and the constraint-coverage relationship, extends to filter definitions we could not evaluate here.

5 Related Work

The analysis of configurable systems is a diverse field of research motivated by the need to manage combinatorial complexity. Basic research has focused on the verification and static analysis of these systems to ensure correctness across all variants [10, 22, 56, 62]. In addition to correctness, the detection of interactions between features is of crucial importance, as different configurations often lead to unexpected deviations in behaviour or performance [3, 6, 59]. To address the scalability issues associated with these analyses, many researchers rely on various sampling strategies to select a manageable subset of configurations [32, 46, 64]. Furthermore, more recent approaches have integrated machine learning to expand the scope of sampling and enable predictive performance modelling [27].

5.1 Analysis and Verification of Configurable Systems

Taxonomies and Strategies. Given the increasing complexity of configurable systems, effective analytical techniques are of crucial importance. Thüm et al. [62] formalised this field by classifying strategies for product-line analysis into three main groups: *product-based* (analysis of individual variants), *feature-based* (analysis of isolated features), and *family-based* (simultaneous analysis of the entire variability model). In their examination of 123 approaches, they distinguish three basic strategies and observe that combined strategies (e.g., feature-family-based) are used to analyse inherently non-compositional properties such as feature interactions.

Static Analysis Approaches. Building on these strategies, Post et al. [55] proposed a meta-programming approach that combines feature models, build rules, and pre-processor logic into a single representation. This enables standard verification tools to verify properties simultaneously across all valid configurations. Their evaluation on a system with over 4 600 features revealed two previously unknown bugs in the kernel's configuration system, surfaced through uncommon configurations that would have required substantially more effort to find if each configuration had been checked individually. Similarly, Beek et al. [10] introduced static analysis techniques for *Featured Transition Systems*, which were formalised by Classen et al. [19]. Beek et al. provided effective algorithms for detecting unreachable transitions, false optional transitions, or hidden deadlocks, proving both the correctness and completeness of their approach.

Efficiency trade-offs. To determine the practical costs of such variability-aware analyses, Rhein et al. [56] compared variability-aware analysis with sample-based approaches using real-world systems such as *OpenSSL*. Their results suggested a trade-off: on average, the variability-aware analysis of a file requires the same amount of time as analysing two individ-

ual variants. Although it is therefore less efficient than checking a single configuration, it is significantly more efficient than pairwise sampling and also offers the additional advantage that warnings are reported for all valid system variants. These verification approaches view the configuration space as something that must be analysed exhaustively for correctness, rather than sampled. However, they provide the starting point for our approach: as soon as an exhaustive analysis is no longer feasible and sampling becomes necessary, the focus shifts to the question of how the quality of a sample can be measured and which coverage metric should be used for this purpose, which is the central question of this thesis.

5.2 Efficient Performance Analysis

While the sections above focus on correctness, other approaches focus specifically on performance. Obtaining accurate performance models typically requires extensive measurements, which is often infeasible [28].

Data-Efficient Learning. To mitigate measurement costs, Guo et al. [29] introduced *DECART*, a data-efficient learning framework. DECART combines machine learning with statistical techniques to build and validate prediction models using only a small sample of configurations, without requiring a separate validation sample.

White-Box Performance Analysis. Moving beyond black-box learning, Velez et al. [65] proposed *ConfigCrusher*, a white-box approach. ConfigCrusher uses static *data-flow* analysis to examine how configuration options affect *control-flow* statements, with only the sections relevant to the measurement being instrumented. Although ConfigCrusher proved to be more accurate than some black-box models, it struggled to scale due to the computational overhead associated with static analysis. To address this issue, Velez et al. [66] introduced *Comprex*, a method that combines local measurements, dynamic taint analysis, and configuration space compression. This enables the creation of accurate models for performance prediction without having to rely on machine learning to extract information from incomplete samples. These performance approaches are based on a representative sample of configurations, but are founded on a predefined notion of what a sample should cover. Our work complements these approaches: rather than building predictive models, we investigate how the choice of coverage metric influences which sampling strategy appears to produce the sample with the best coverage.

5.3 Foundations of Feature Interactions

Definition and History. Feature interactions happen when the presence of other features alters the influence of a particular feature on a non-functional attribute (e.g., performance) [59]. The concept, historically associated with telecommunications [16], has expanded to fields such as internet applications, automotive systems, and computational biology [3]. Calder and Miller [17] identified interactions between properties through the pairwise analysis of properties and temporal properties represented in linear temporal logic, using model checking in conjunction with symmetry reduction and bisimulation to manage the resulting state-space explosion.

Causal Reasoning. Recently, research has focused on identifying interactions through causal reasoning. Dubslaff et al. [22] introduced the concept of *feature causality*, inspired by Halpern and Pearl's actual causality [30]. Feature causality, which operates at the configuration level, uses *counterfactual reasoning* to determine which features and interactions are actually responsible for a property. By linking feature causality with prime implicants, they provided efficient algorithms for computing *blame* and *responsibility*. In a follow-up paper, Dubslaff et al. [23] further refined this approach to include quantitative measures of causal effects and reasoning under uncertainty, and demonstrated its effectiveness on real-world systems.

Evolutionary Perspective. Finally, static analysis alone is not sufficient for modern systems. Kolesnikov [39] conducted an empirical study of the nature and causes of feature interactions, as well as their influence on the performance of real-world configurable systems. The study showed that performance interactions are only externally observable through measurement and cannot be fully identified from a system's control flow alone. Sattler [57] extends this perspective by explicitly incorporating development context into the analysis of configurable systems. Sattler argues that configuration-specific performance regressions and non-functional-property interactions are difficult to interpret in an evolving system unless they are related to their development context. This motivates a shift towards a development-oriented approach that contextualises analysis results within the evolution of the system.

5.4 Feature Interaction Detection

On this foundation, the detection of functional interactions aims to identify instances where such dependencies lead to unexpected behaviour, faults, or performance deviations that are not visible when features are analysed in isolation.

Correctness-Oriented Detection. An approach focused on the automatic detection was proposed by Apel et al. [6], who used *feature-aware verification* to detect feature interactions in configurable systems. Instead of analysing each product variant individually, *feature-aware verification* explicitly considers variability and can thus avoid some of the redundant verification caused by product similarities. In related work, Apel et al. [5] investigated whether feature-based specifications, which describe each feature in a modular way without reference to others, can detect interactions that, by their very nature, cross feature boundaries. In a study of 10 feature-oriented systems, the majority of interactions were detectable, although workarounds were required for some specifications where clean modularisation was not possible.

Performance-Oriented Detection. Feature interactions were also investigated from a performance perspective. Siegmund et al. [59] proposed an automated approach for detecting performance-relevant interactions between features. Their work is particularly relevant for performance analysis of configurable systems because it treats interactions as effects that influence non-functional properties, rather than only correctness. However, detecting such interactions by measuring many configurations is expensive, which motivates more efficient and lightweight approaches.

Lightweight Dynamic Detection. To avoid having to perform a comprehensive analysis of all configurations, researchers have proposed lightweight methods for identifying interactions. In this context, lightweight means that the approach attempts to learn or identify interactions without scanning the entire configuration space. Soares [60] introduced *VarXplorer*, which uses variational trace to monitor interactions in the control and data flow of highly configurable systems. The identified interactions are presented as *feature-interaction graphs*, which help developers understand how features influence each other during program execution.

Dynamic Interaction Inference. Nguyen et al. [49] introduced *iGen*, a dynamic approach to deriving interactions. iGen executes selected configurations and observes which configurations cover certain program elements and which do not. Based on this information, iGen generates logical formulae that describe the interactions between the configuration options. This process continues until no more interactions can be identified. However, iGen is limited to certain types of formulae, which restricts its ability to model more complex interactions in real-world systems. To address this limitation, Nguyen et al. [48] introduced *GenTree*. Similar to iGen, GenTree learns interactions from observed program behaviour, but uses decision trees to derive logical formulae that explain the effect of configuration options on code coverage. This enables GenTree to represent more complex interaction structures while still requiring only a small portion of the entire configuration space.

5.5 Sampling Configurable Systems

Since it is impossible to cover all valid configurations due to the combinatorial explosion [15, 70], researchers have developed various strategies for selecting a representative subset of configurations. These strategies can be divided into two categories: those that focus on statistical distribution and those that focus on structural coverage.

Distribution-Based Approaches. These strategies aim to sample configurations that are uniformly or stochastically spread across the valid space. Plazar et al. [54] pointed out the challenges that come with using *SAT-based uniform samplers* for large, complex feature models. Recent work has investigated *BDD-based uniform samplers* as an alternative to SAT-based approaches. However, it has become apparent that their practical scalability depends on whether the BDD representation can be constructed efficiently [36]. As an alternative to strict uniformity, Kaltenecker et al. [37] introduced *distance-based sampling*. Unlike strategies that maximise the coverage of structural interactions, this approach uses a distance metric to spread configurations diversely across the solution space. In an evaluation of ten real-world systems, Kaltenecker et al. showed that distance-based sampling outperforms traditional strategies in terms of predictive accuracy (mean error rate) and robustness (variance) for samples of medium to large size.

Interaction-Coverage Approaches. A distinct class of strategies, often referred to as *coverage-oriented*, aims to systematically cover certain structural combinations of features. The best-known approach is *t-wise sampling*, which ensures that every combination of t features occurs at least once [68].

Fault-Based Heuristics. In addition to the standard combinatorial coverage, researchers have adapted their sample selection to specifically capture certain error patterns. Abal et

al. [1] classified the conditions under which errors occur into classes such as *some-enabled* (where certain features must be enabled) and *some-enabled-one-disabled* (where certain features can be enabled and exactly one must be disabled). Based on these findings, Medeiros et al. [46] compared ten modern sampling algorithms in terms of fault detection and sample size. They evaluated strategies such as *one-enabled*, in which a solver selects configurations in which a single feature is enabled at a time, and *most-enabled-disabled*, which targets configurations in which the greatest number of features are enabled or disabled. Their results showed how effective these heuristics are at detecting feature-specific errors compared to broader sampling methods.

Search-Based and Multi-Criteria Sampling. Although the standard t-wise sampling method is efficient for testing purposes, it has its limitations. Krieter et al. [41] found that many approaches fail to take into account how feature combinations are mapped to the actual solution space, and proposed a new criterion that incorporates presence conditions. Furthermore, achieving high t-wise coverage ($t \geq 3$) is computationally expensive. To address this, Pett et al. [53] proposed *MulTi-Wise Sampling*, which assigns higher t-values to known critical feature groups while reducing the burden for less important ones. Another approach to addressing the limitations of t-wise sampling was introduced by Henard et al. [32] and is based on a search-based method that can be used to generate product configurations for large SPLs. As such, it offers a scalable and flexible option to existing methods and prioritisation algorithms for arbitrary sets of product configurations. Both approaches use a similarity approach. To test their approach, they conducted an experiment, in which Henard et al. compared their new approach with t-wise sampling. For the experiment, Henard et al. generated 100 artificial feature models and used 14 real feature models ranging from $t = 2$ to $t = 6$. The approach developed by Henard et al. made it possible to process the most comprehensive feature models, as exemplified by the Linux kernel ($\approx$ 7 000 features, 200 000 constraints, and 8.71×10^{21} valid 6-sets), achieving a coverage of up to 90.671% with 1 000 configurations.

Incremental and Model-Based Strategies. Finally, recent work has moved towards incremental generation to avoid the redundancy of regenerating test suites from scratch. Uzuncaova et al. [63] used the *AHEAD* methodology [9] to map feature specifications to transformations over test suites, enabling incremental generation of new tests by refining tests generated for existing products. Extending this model-based perspective, Lochau et al. [44] provided a mapping between feature models and reusable statechart test models, allowing for the verification of control and data flow coverage criteria within a pairwise framework. In further work, Lochau et al. [45] introduced a delta-oriented test framework in which product variants are tested sequentially. By making targeted use of the commonalities between successive products, this approach reduces redundancies while ensuring stable test coverage across the entire sample set.

5.6 Coverage Metrics for Feature Interactions

While interaction detection techniques aim to identify interactions relevant to behaviour or performance, sample-based studies often evaluate samples on the basis of structural interaction coverage. However, there is no standard definition of such coverage. Böhm et

al. [15] showed that different t-wise coverage metrics take into account different subsets of feature interactions, for example by excluding mandatory features, dead features, atomic sets, or parent-child interactions. As a result, the same sample can receive different coverage values depending on the chosen metric.

5.7 Performance Modelling & Prediction

Machine learning serves as a complementary extension to sampling, using selected measurement values to derive performance models for the entire system. Siegmund et al. [58] pioneered the use of regression methods to predict performance based on small training datasets. In a subsequent comparison, Grebhahn et al. [27] evaluated techniques such as *Random Forest*, *CART*, *Support Vector Regression*, and *Kernel Ridge Regression*. They concluded that both the learning method and the sampling strategy have a significant impact on predictive accuracy, although there is no single combination that delivers the best performance across all systems.
In recent years, the field has shifted its focus to deep learning in order to better capture complex, non-linear interactions between features, a trend that has been investigated in detail by Gong et al. [26]. Their review focuses exclusively on deep learning for performance modelling of configurable software. Based on 1 206 papers retrieved from six indexing services, Gong et al. identified and analysed 99 primary studies. Their findings summarise key statistical metrics and propose a taxonomy, while also discussing the advantages and disadvantages, as well as suitable application scenarios of deep learning techniques across the entire modelling pipeline, including configuration data preparation, model construction, evaluation, and application to tasks related to software configuration. While these approaches use sampled measurements to make predictions about the performance of the entire system, they treat the sampling strategy and its coverage as given. This thesis instead focuses on the evaluation of the sampling stage itself and investigates whether the coverage metric used to assess a strategy influences the conclusions drawn from it.

6

Conclusion

In this thesis, we conducted an empirical study on 21 real-world configurable software systems to investigate whether the choice among the eight coverage metrics influences the coverage values generated by random, distance-based, and solver-based sampling, and to determine the correlation between a system's constraint strength and its achieved coverage. For each system, we used t-wise sampling at sample sizes $t_s = 1, 2, 3$ to determine the sizes of the sample sets obtained through our chosen sampling strategies, and we computed coverage at coverage strengths $t_c = 1, 2, 3$. We then applied Spearman's rank correlation coefficient and the Kruskal–Wallis test to measure the effect of the selected coverage metrics. Our results show that, while the eight metrics examined throughout this thesis correlate positively with one another, they are not interchangeable. This becomes clear in scenarios with a larger sample size and higher coverage, where the differences between the metrics increase. This effect is most evident in distance-based sampling, the strategy that is most sensitive to the choice of metric. Since the metrics are not interchangeable, the evaluation of a sampling strategy is not metric-neutral: the same sampler may appear more or less favourable depending on the metric used for evaluation, particularly under more demanding evaluation conditions.

The choice of metric has little effect on the ranking of the three sampling strategies. Random sampling consistently achieves the highest coverage values, followed by distance-based sampling and, finally, solver-based sampling. While the most demanding conditions can shift this order in some respects, the ranking remains largely unchanged across the eight coverage metrics.

A similar pattern appears in our third and final experiment. The metrics have almost no influence on the correlation between the constraint strength and the coverage values. Instead, the sample size and the coverage strength have a far greater influence on the correlation. The practical result is that coverage values cannot be compared across systems without taking the coverage strength into account: a higher value in a heavily constrained system does not indicate a better sample, but simply reflects its smaller interaction space.

Overall, our results point to a common conclusion: in all three experiments, the nature of a sample's coverage is determined far more by the sample size and coverage strength than by the coverage metric. Although the metric determines the absolute coverage value assigned to a given sample, it does not alter either the ranking of the sampling strategies or the relationship between coverage and the constraint strength. When evaluating our chosen sampling strategies, therefore, the choice of coverage metric is less crucial than the conditions under which coverage is measured.

A

Appendix

In this chapter, we provide supplementary material for the evaluation. The appendix is structured by research question. For **RQ1**, we include additional visualisations of metric agreement and signed coverage differences between Default and the remaining metrics. For **RQ2**, we provide further strategy-specific comparisons and distributional views of the coverage values. For **RQ3**, we provide the metric-wise scatter plots relating constraint strength to achieved coverage.

A.1 Supplementary Results for RQ1

This section complements the results for **RQ1** by providing additional visualisations of metric relationships. It first shows an overall summary of agreement between metric pairs, then presents strategy-specific agreement heatmaps, and finally reports the signed pointwise coverage differences between Default and the remaining metrics.

a.1.1 Overall Metric Agreement

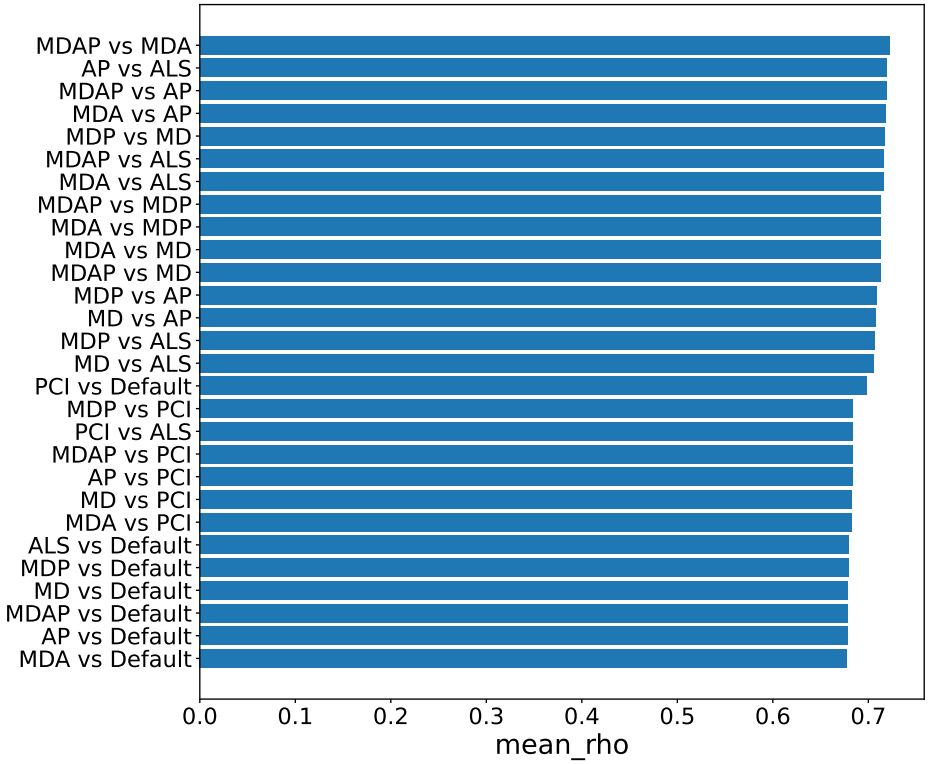
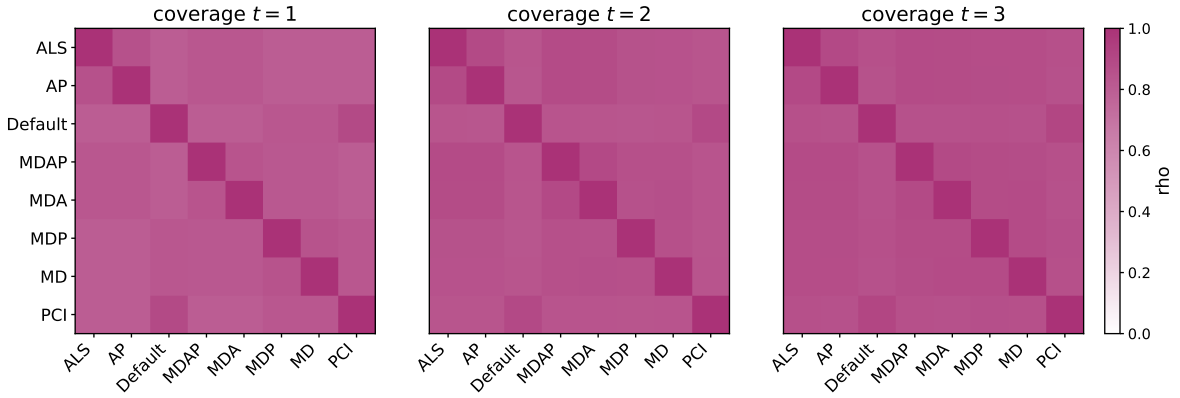


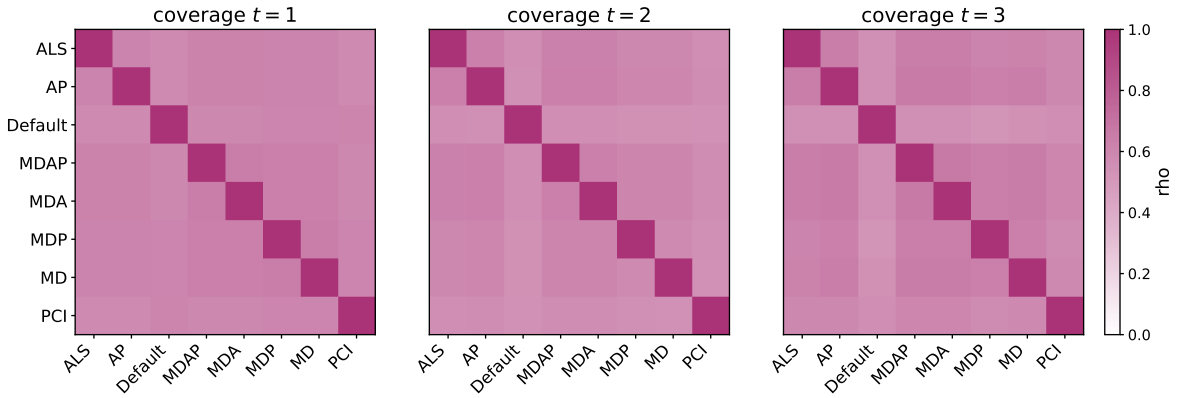
Figure A.1: Average Spearman correlation between metric pairs, ranked across all systems, sampling strategies, sample size, and coverage strengths.

a.1.2 Metric Agreement by Sampling Strategy

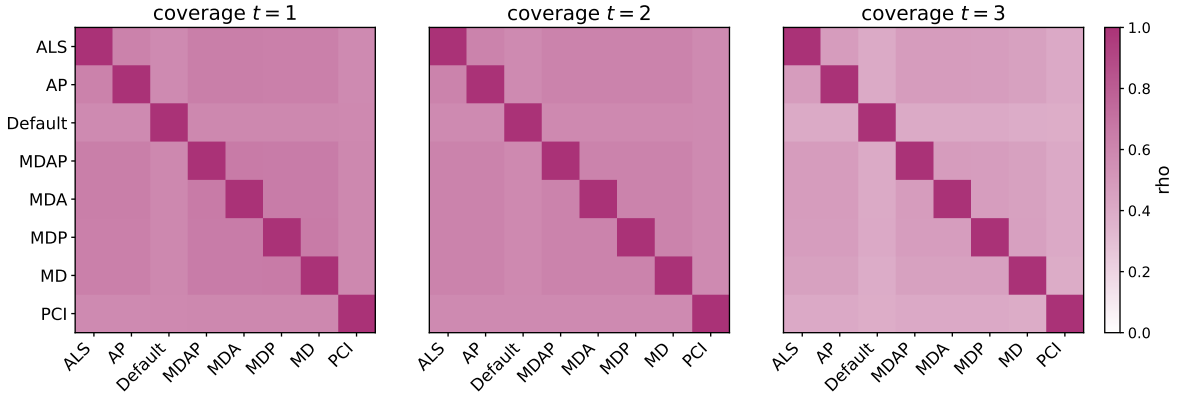
This subsection provides additional heatmaps of the Spearman correlation coefficients for the three sampling strategies separately. The figures show how the agreement between metrics changes across coverage strengths t_c and sample size t_s.



(a) $t_s = 1$

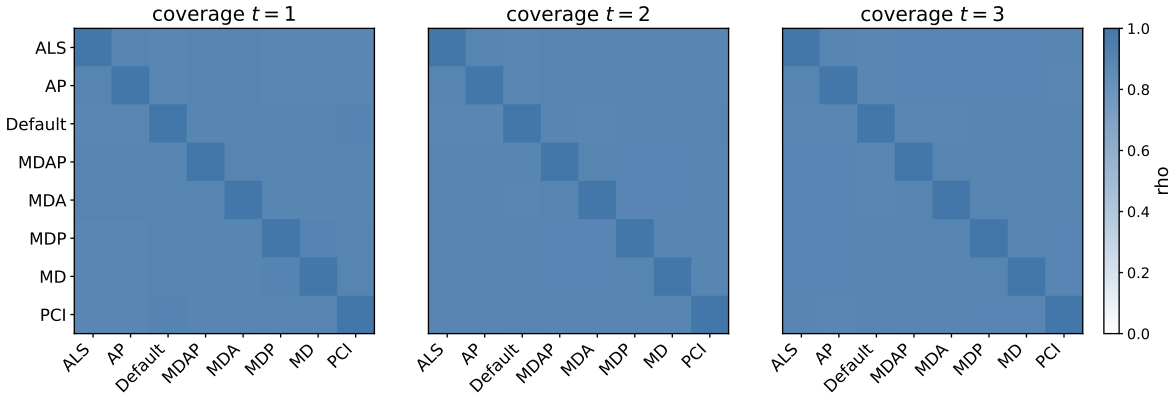


(b) $t_s = 2$

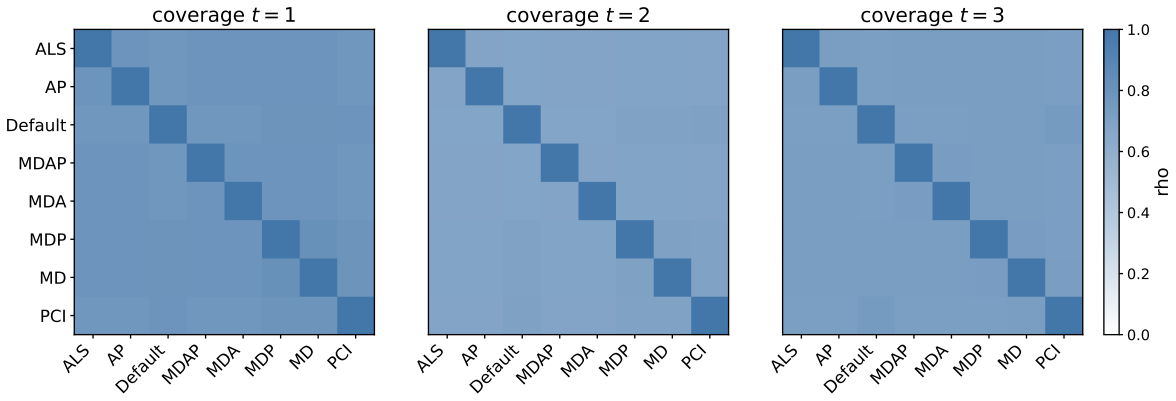


(c) $t_s = 3$

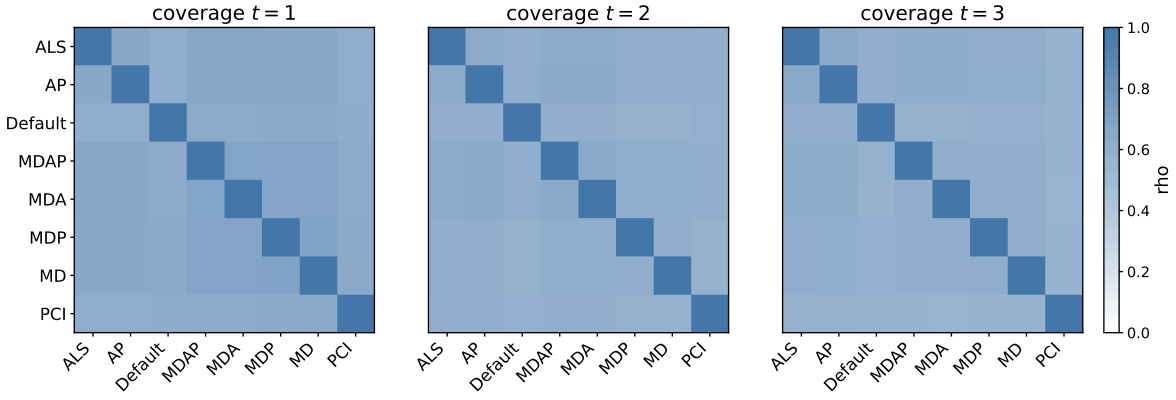
Figure A.2: Metric agreement for solver-based sampling across sample size t_s.



(a) $t_s = 1$

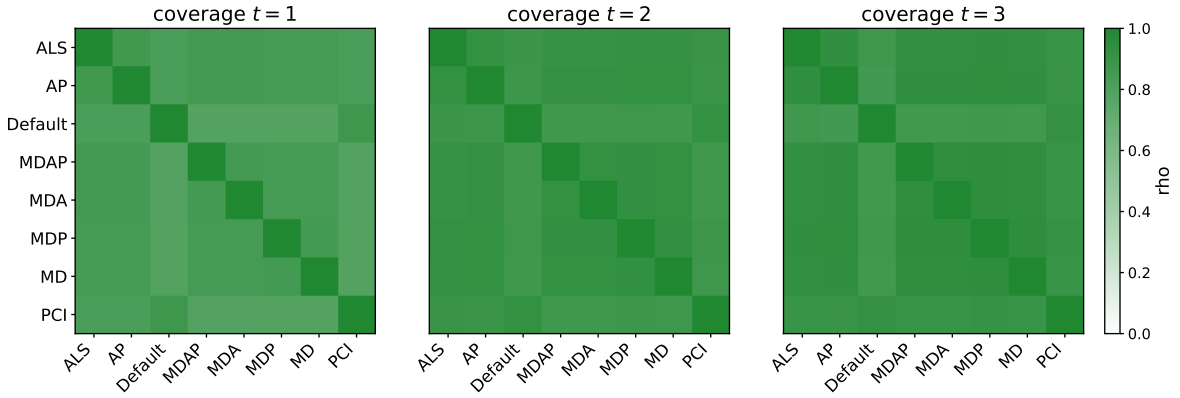


(b) $t_s = 2$

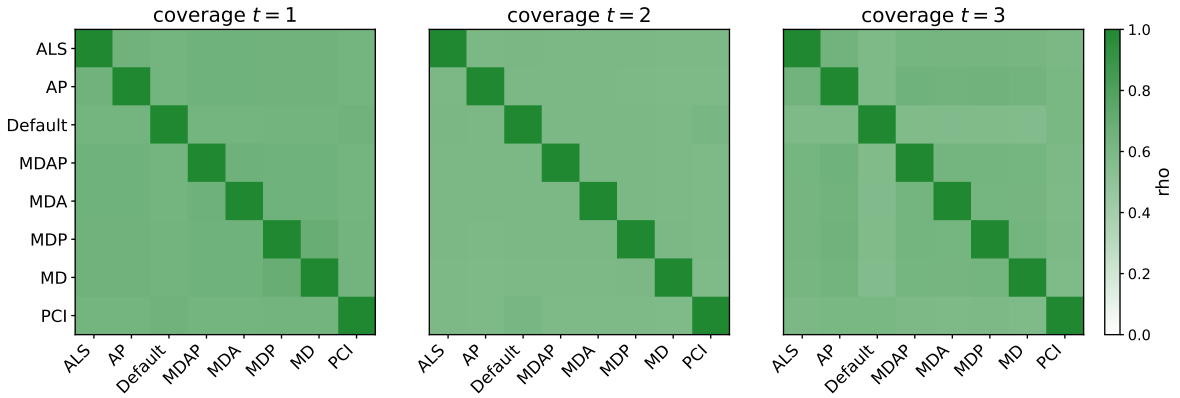


(c) $t_s = 3$

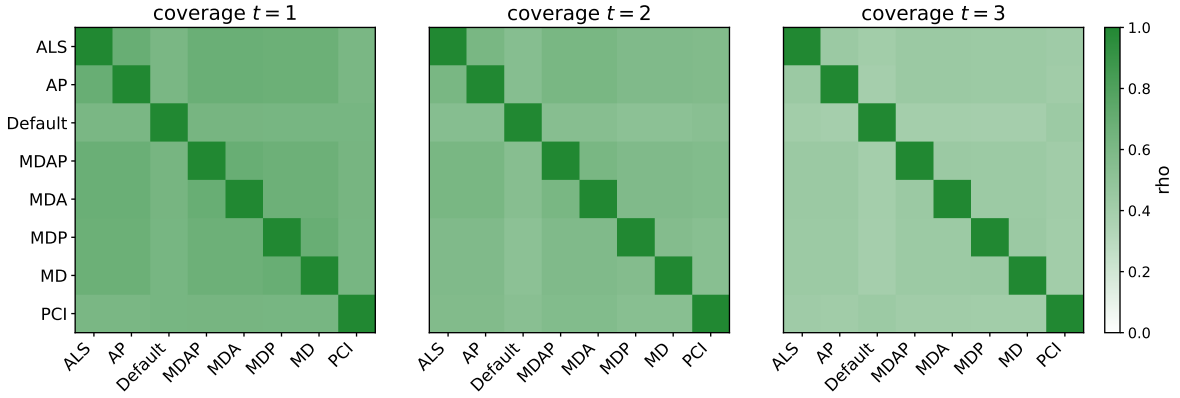
Figure A.3: Metric agreement for random sampling across sample size t_s.



(a) $t_s = 1$



(b) $t_s = 2$



(c) $t_s = 3$

Figure A.4: Metric agreement for distance-based sampling across sample size t_s.

a.1.3 Signed Coverage Differences Relative to Default

This subsection shows the distribution of signed pointwise coverage differences between Default and the remaining metrics. Positive values indicate that Default yields higher

coverage values, while negative values indicate that the respective comparison metric yields higher coverage values.

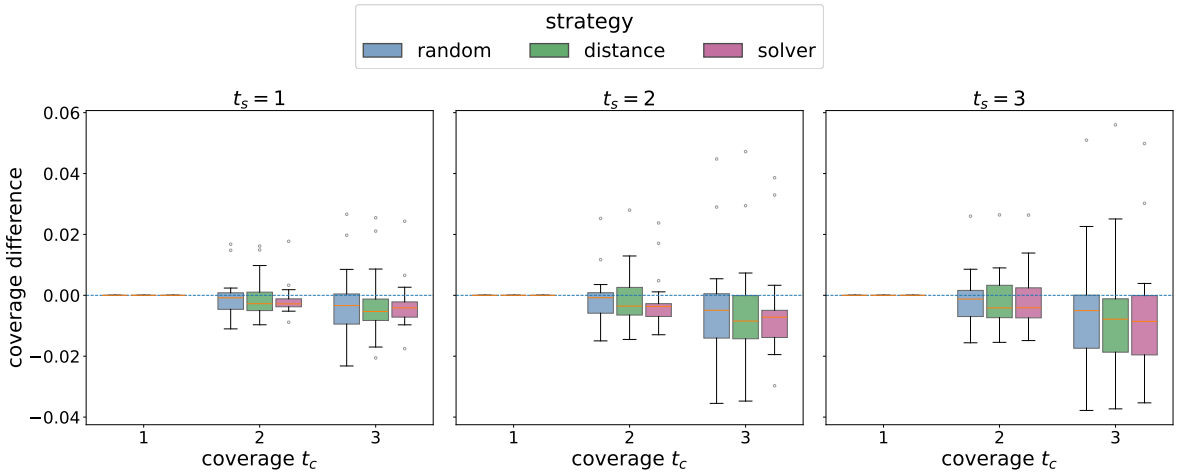


Figure A.5: Distribution of signed coverage differences between DEFAULT and PCI.

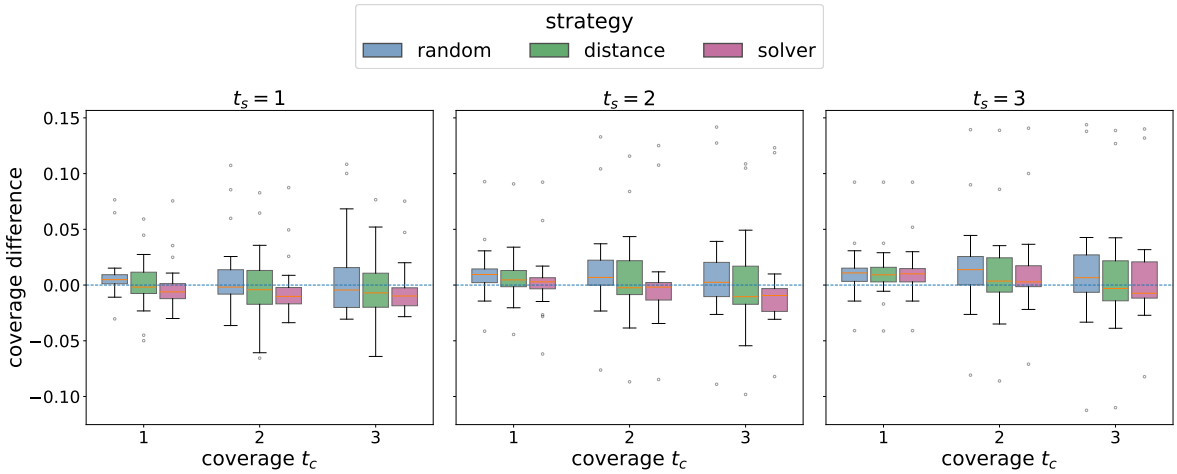


Figure A.6: Distribution of signed coverage differences between DEFAULT and MD.

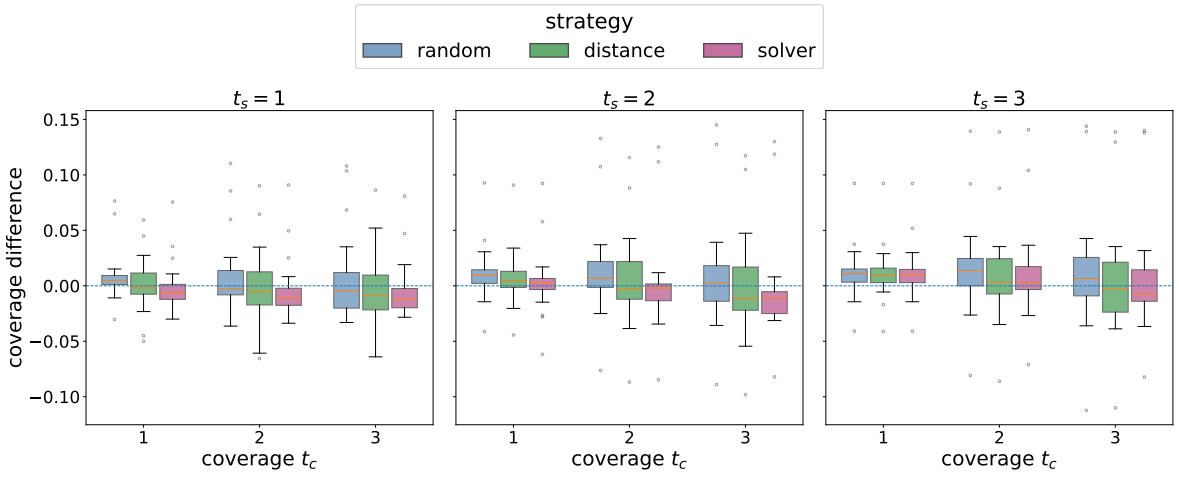


Figure A.7: Distribution of signed coverage differences between DEFAULT and MDP.

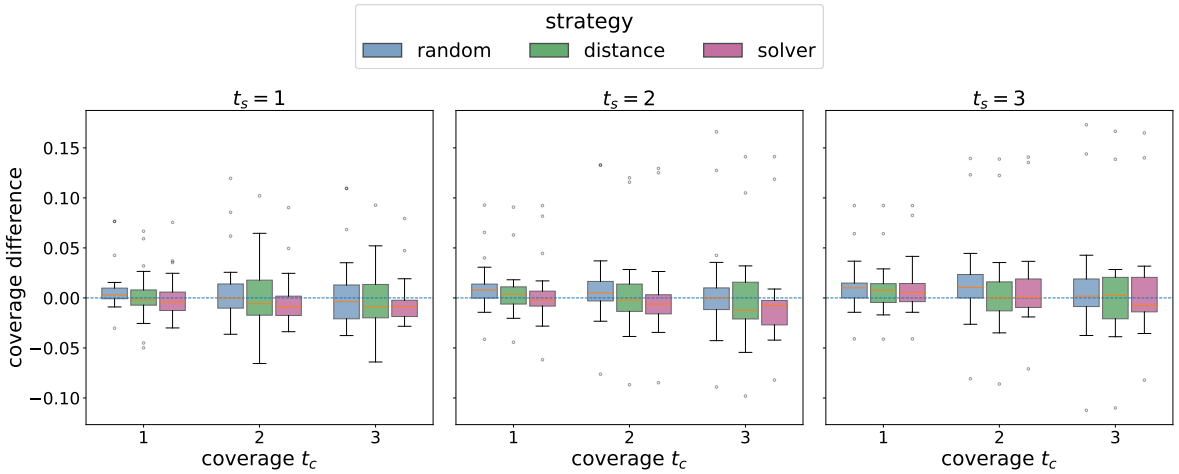


Figure A.8: Distribution of signed coverage differences between DEFAULT and MDA.

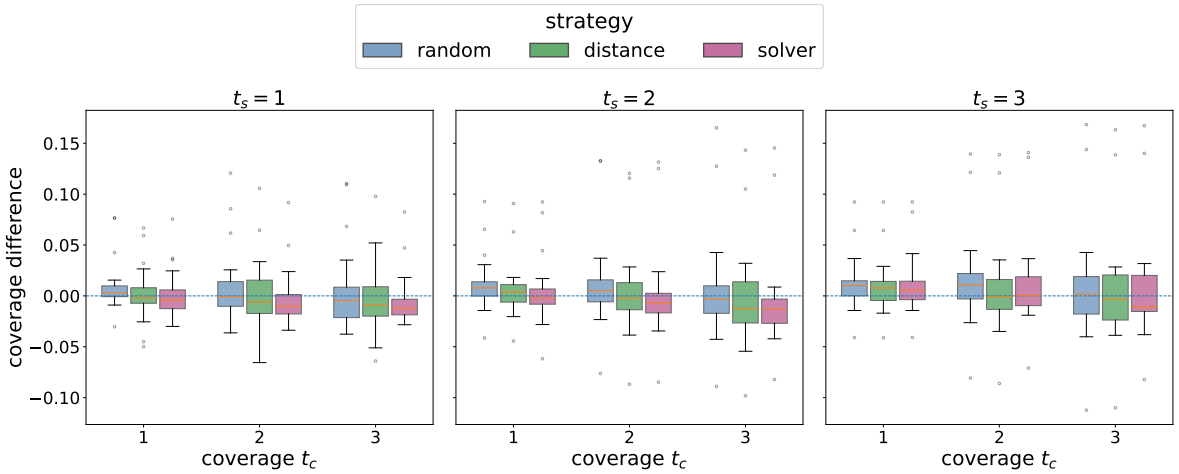


Figure A.9: Distribution of signed coverage differences between DEFAULT and MDAP.

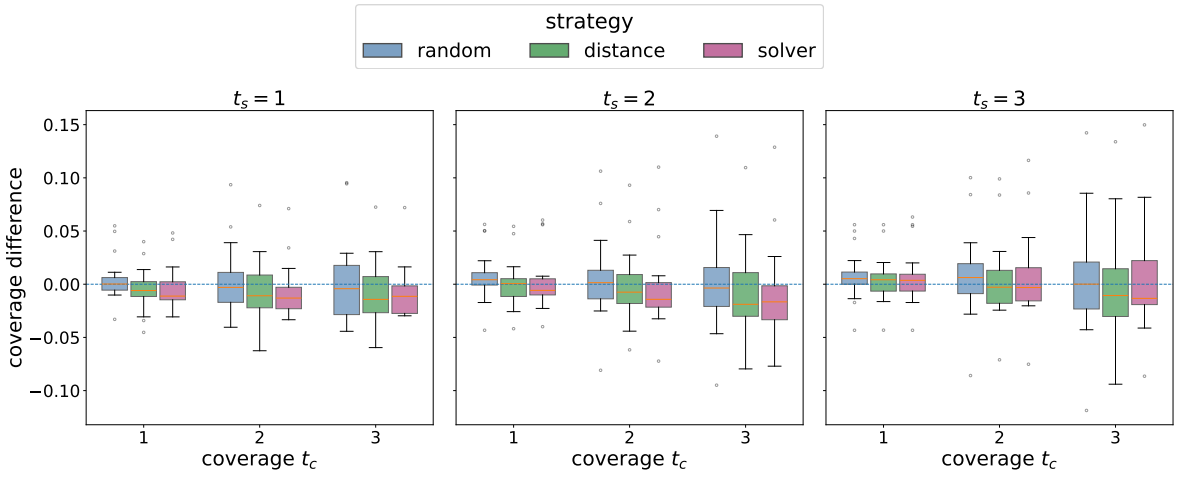


Figure A.10: Distribution of signed coverage differences between DEFAULT and AP.

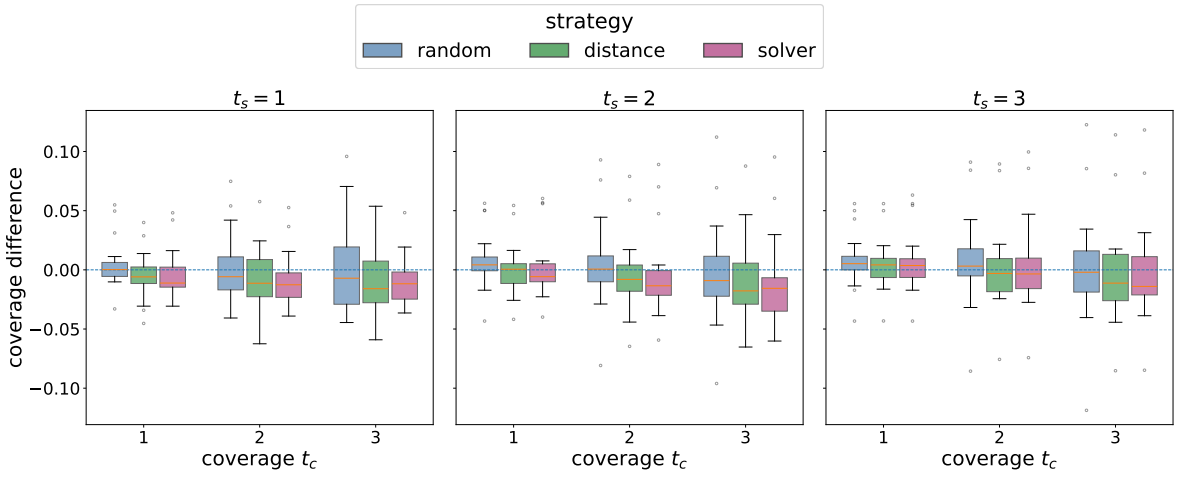


Figure A.11: Distribution of signed coverage differences between DEFAULT and ALS.

a.2 Supplementary Results for RQ2

This section complements the results for **RQ2** by providing additional strategy-specific comparisons and distributional views of the coverage values. It includes box plots of the coverage distributions and full orderings of strategies within settings.

a.2.1 Coverage Distributions by Metric and Strategy

This subsection shows the full distribution of the coverage values for each metric and sample size, grouped by sampling strategy.

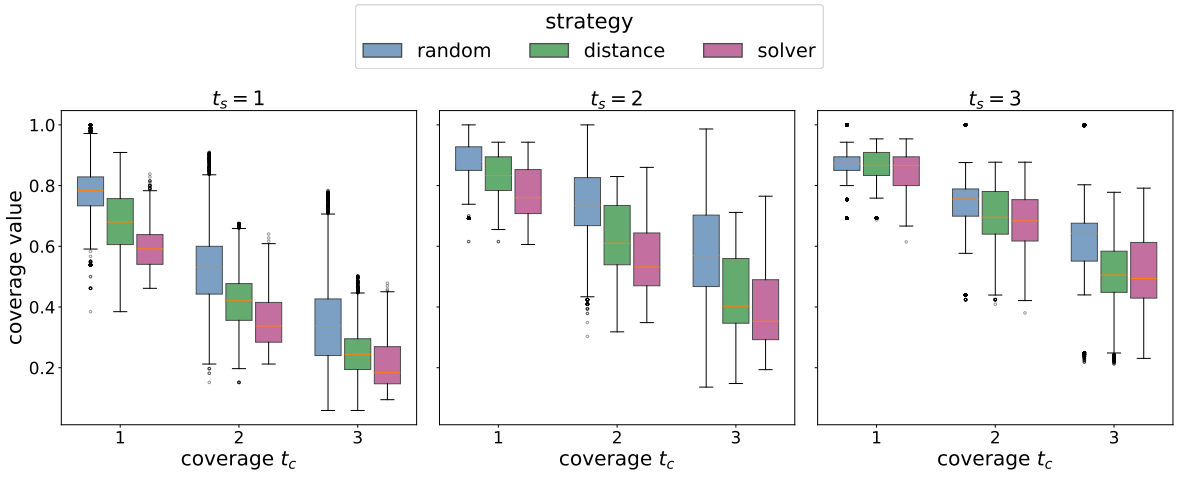


Figure A.12: Coverage distributions for PCI, grouped by sampling strategy.

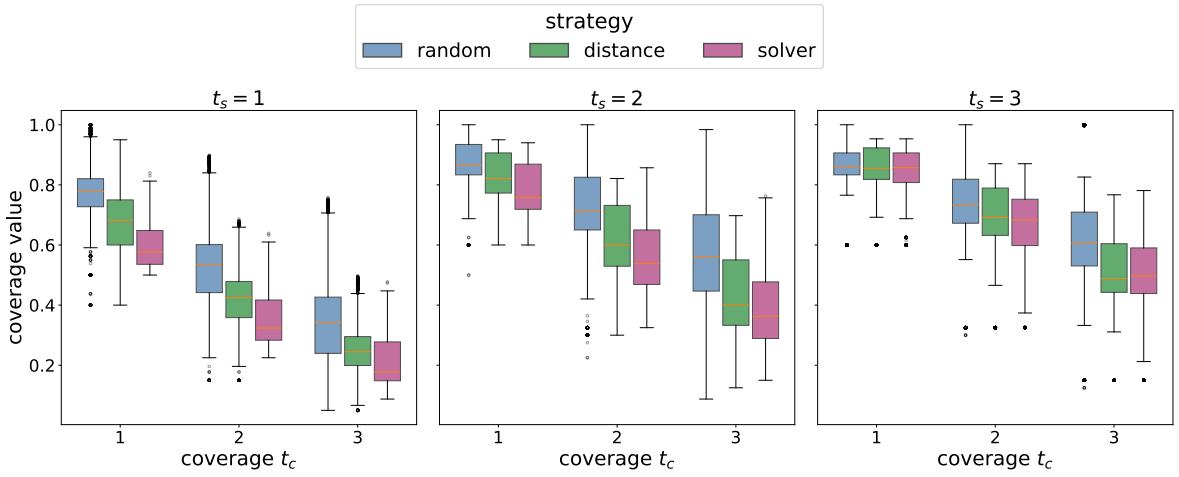


Figure A.13: Coverage distributions for MD, grouped by sampling strategy.

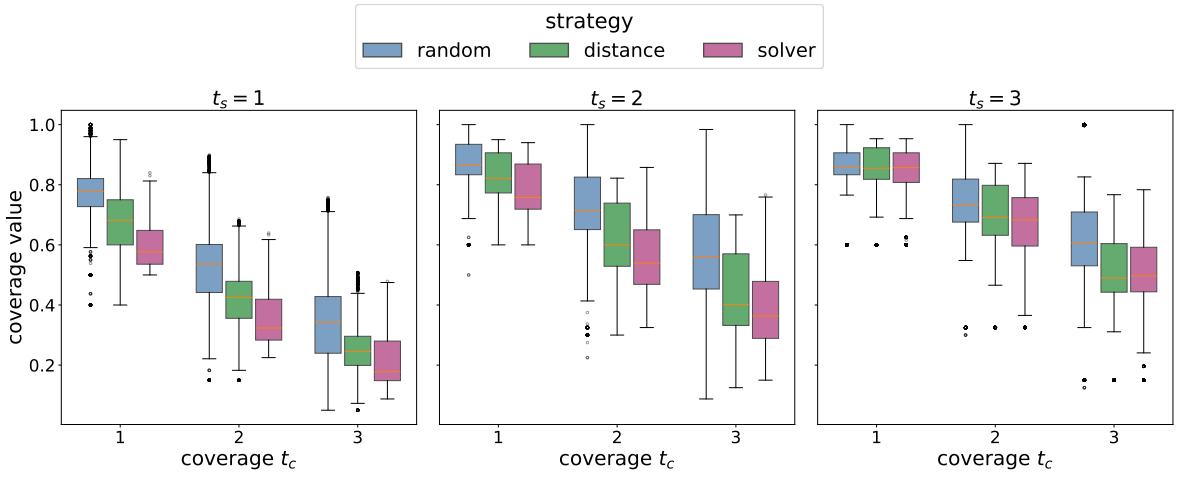


Figure A.14: Coverage distributions for MDP, grouped by sampling strategy.

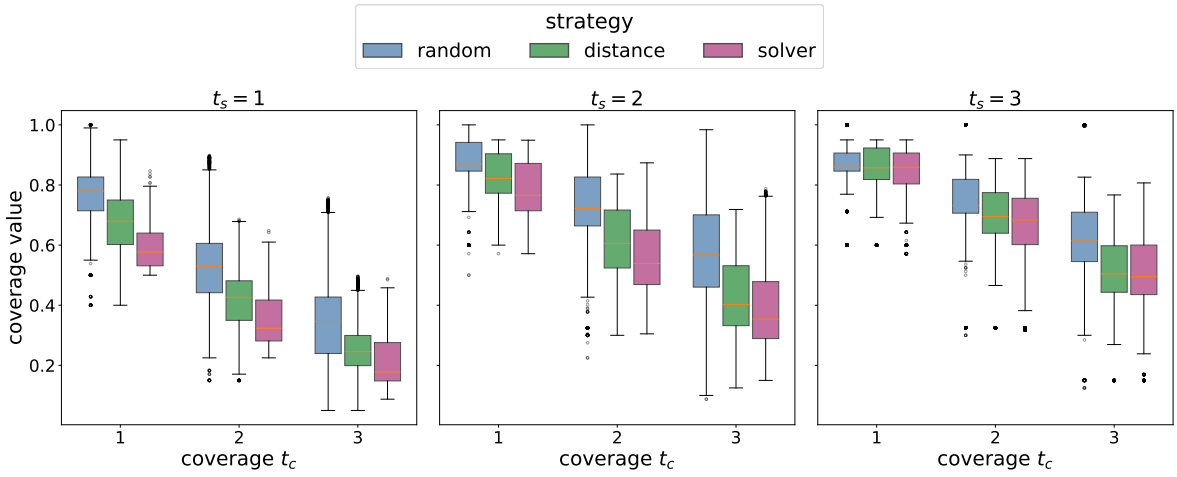


Figure A.15: Coverage distributions for MDA, grouped by sampling strategy.

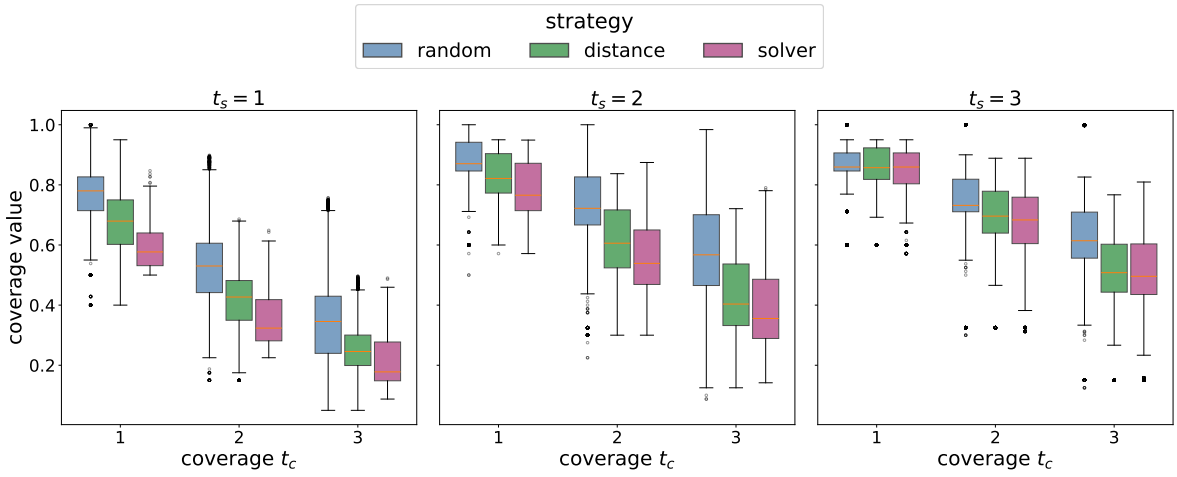


Figure A.16: Coverage distributions for MDAP, grouped by sampling strategy.

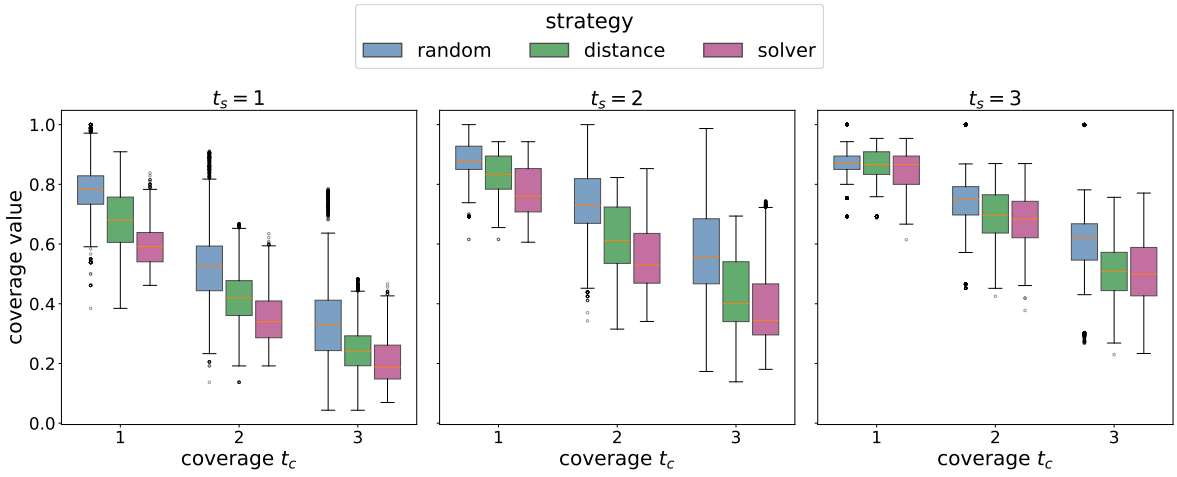


Figure A.17: Coverage distributions for DEFAULT, grouped by sampling strategy.

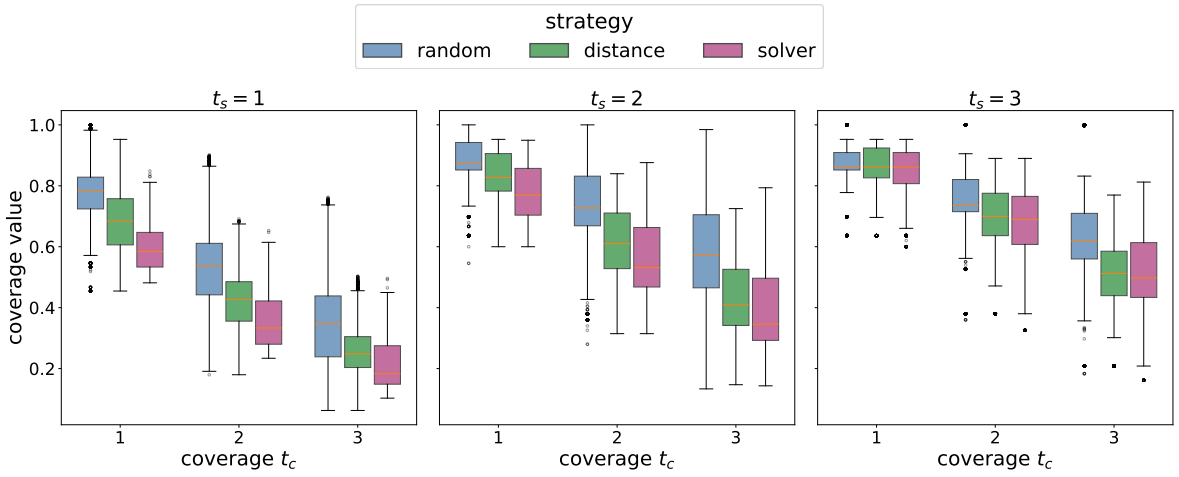


Figure A.18: Coverage distributions for AP, grouped by sampling strategy.

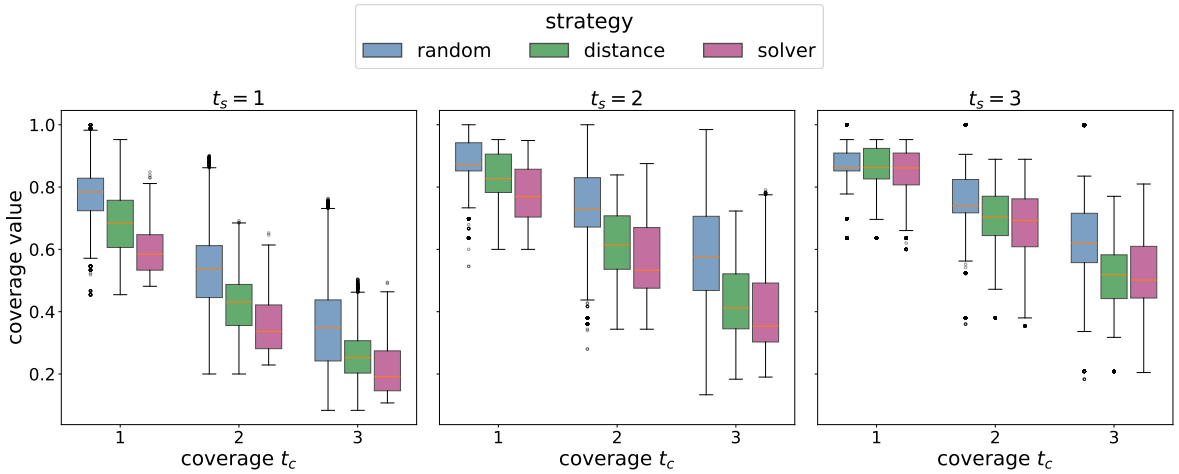


Figure A.19: Coverage distributions for ALS, grouped by sampling strategy.

a.2.2 Full Ordering of Strategies Within Settings

This subsection shows the complete ordering of all strategy-setting combinations for each metric. The plots provide a more fine-grained view than the aggregated win counts by showing where each strategy appears in the full ranking.

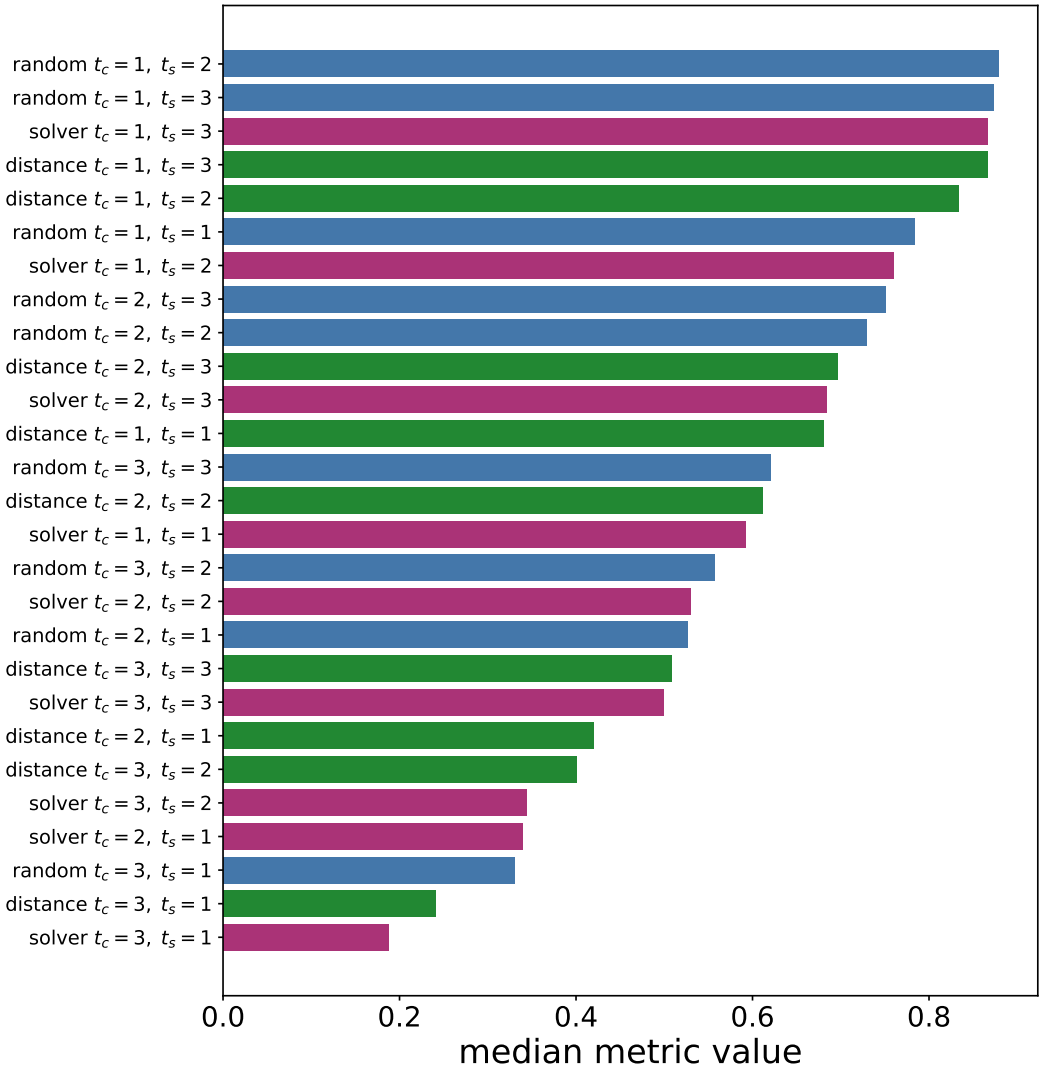


Figure A.20: Full ordering of strategy-setting combinations for Default.

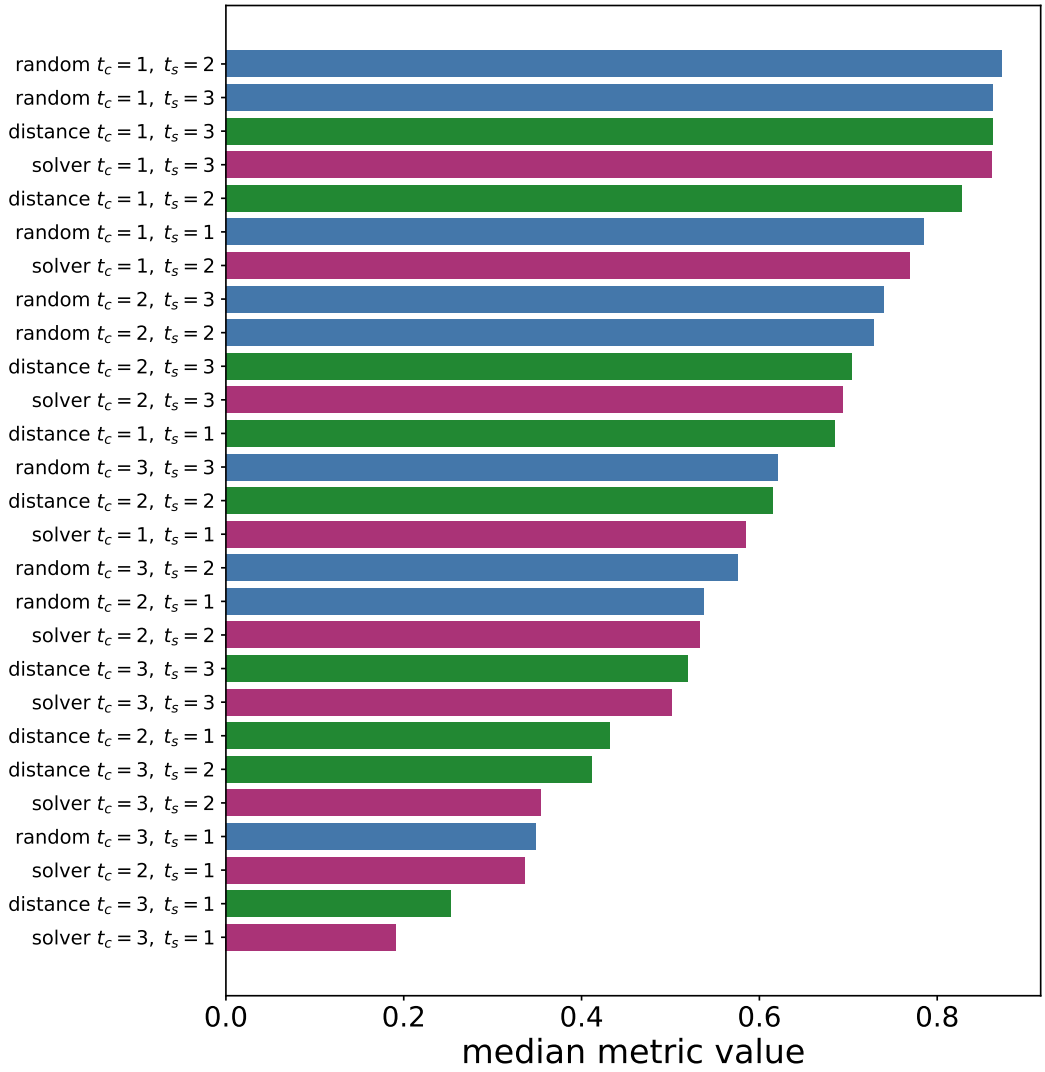


Figure A.21: Full ordering of strategy-setting combinations for ALS.

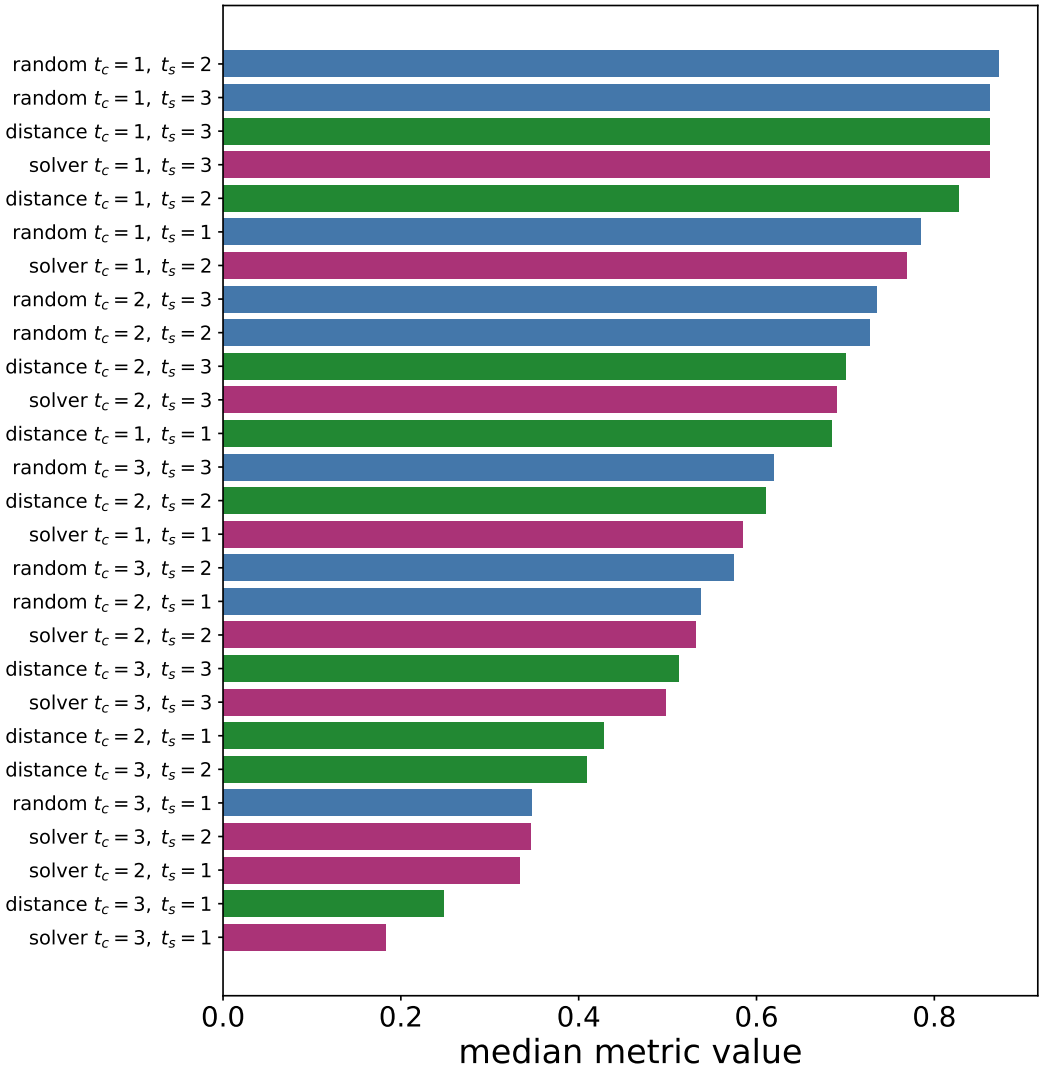


Figure A.22: Full ordering of strategy-setting combinations for AP.

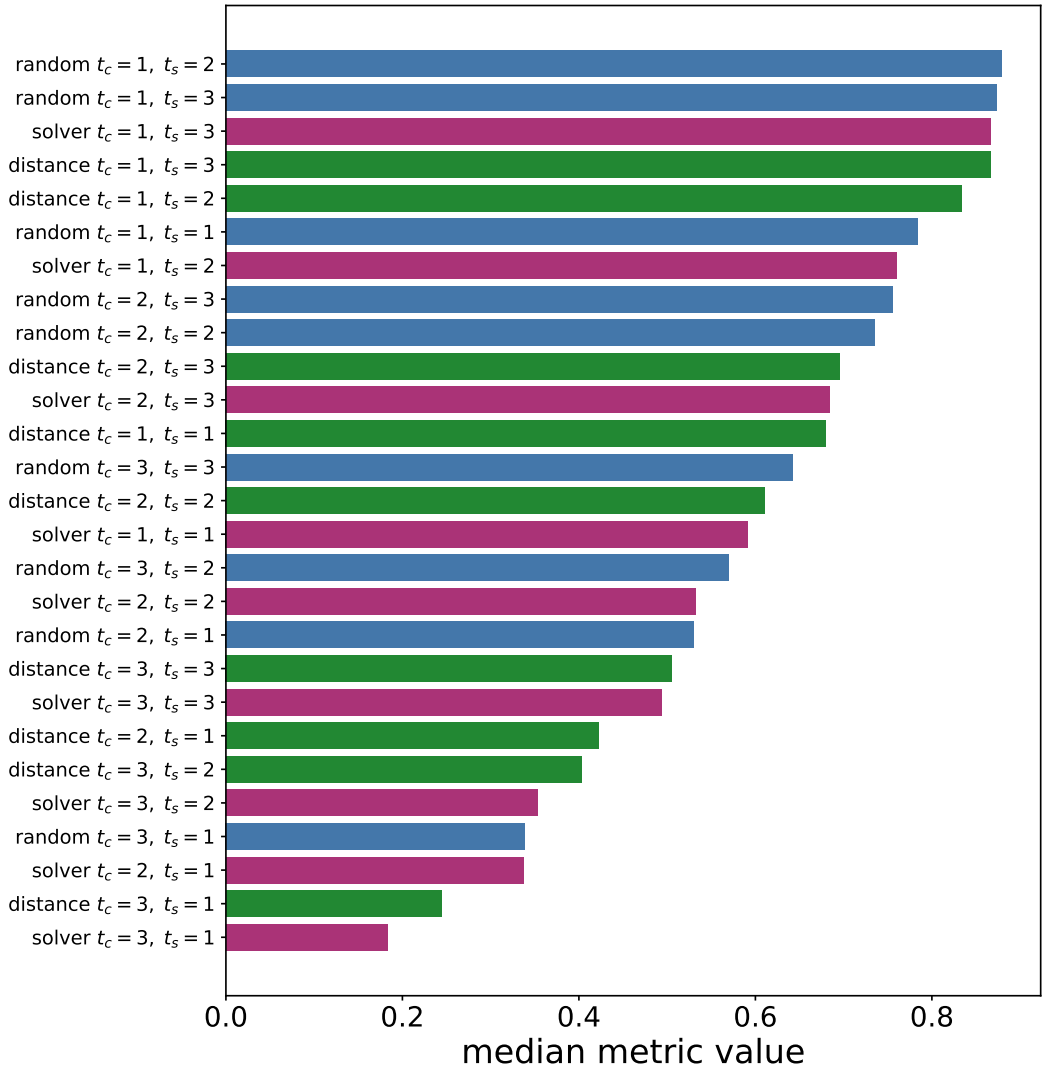


Figure A.23: Full ordering of strategy-setting combinations for PCI.

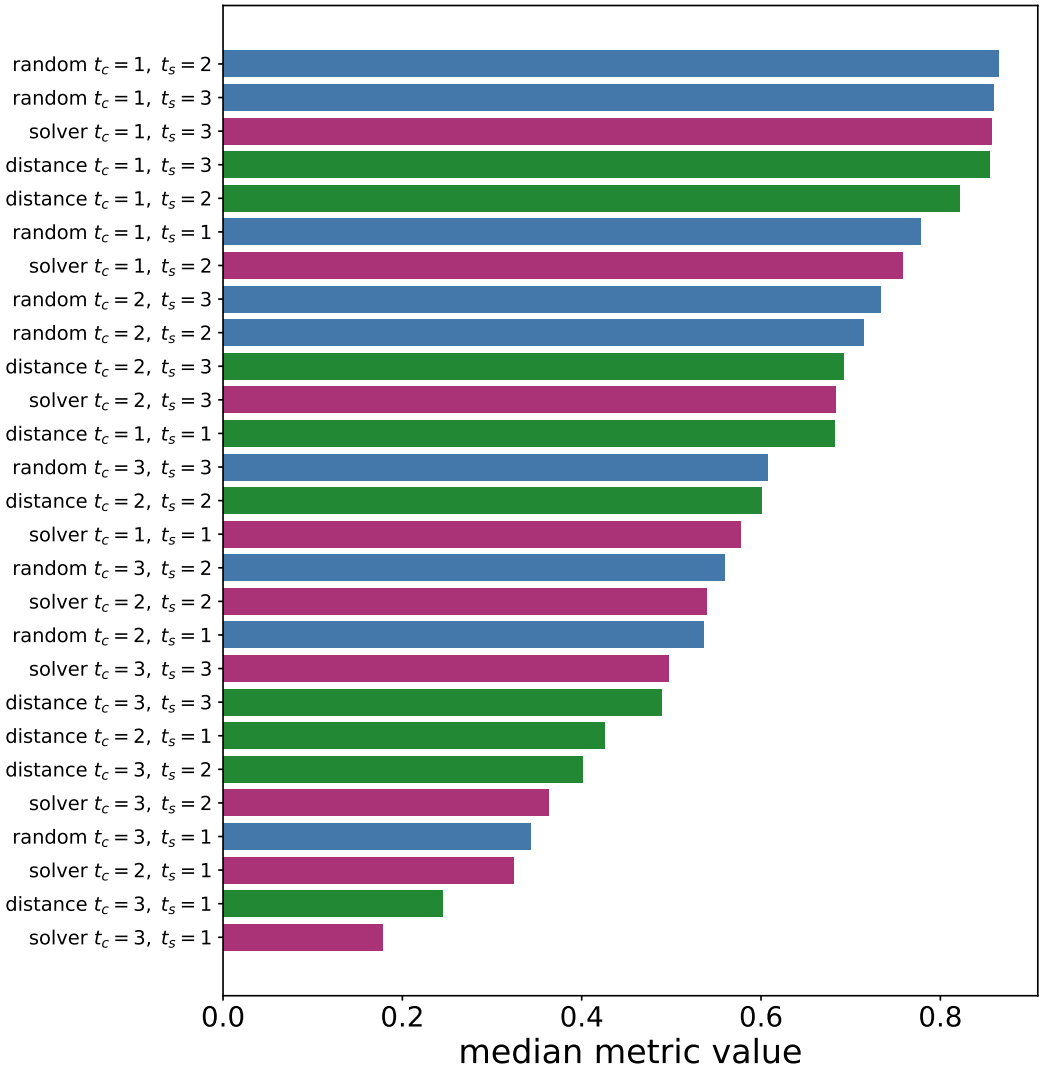


Figure A.24: Full ordering of strategy-setting combinations for MD.

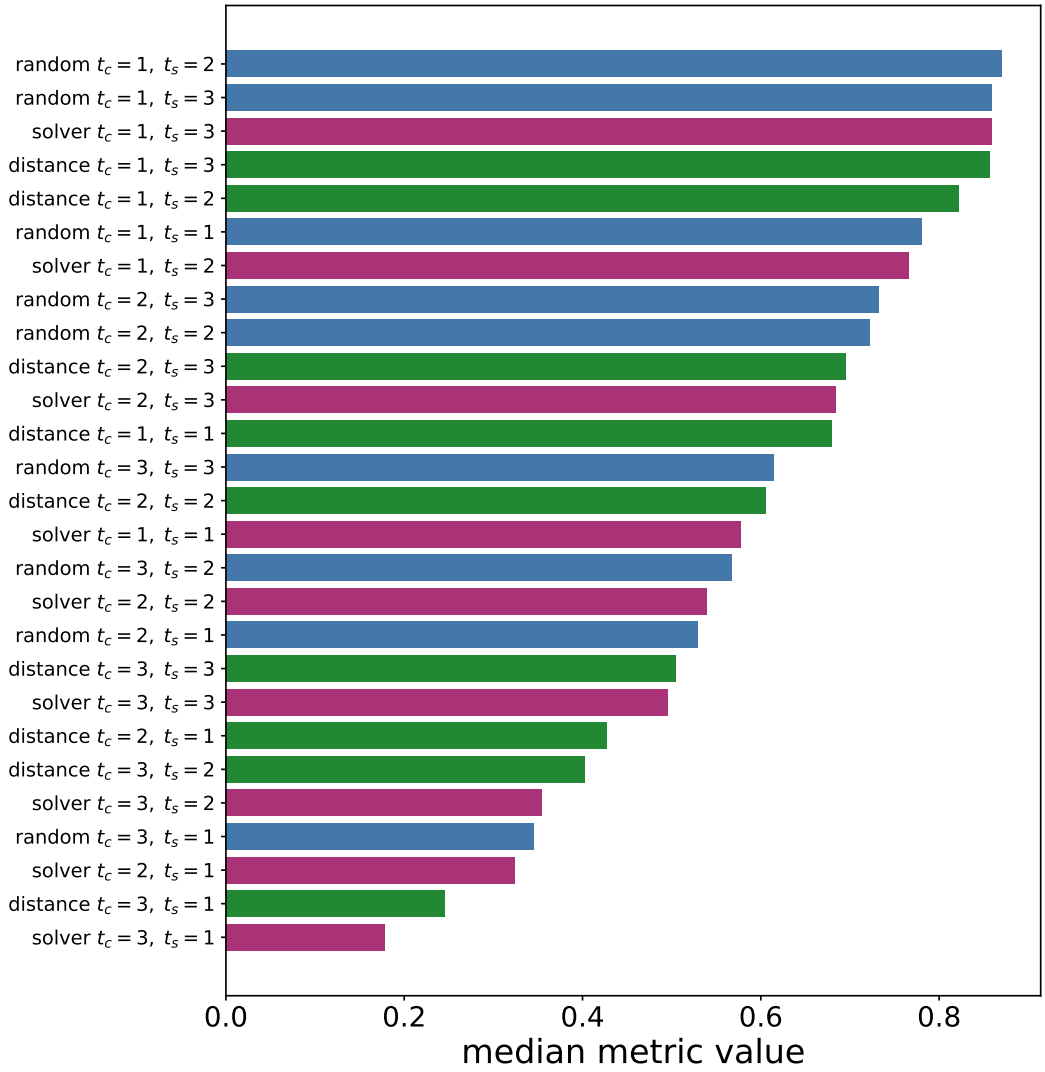


Figure A.25: Full ordering of strategy-setting combinations for MDA.

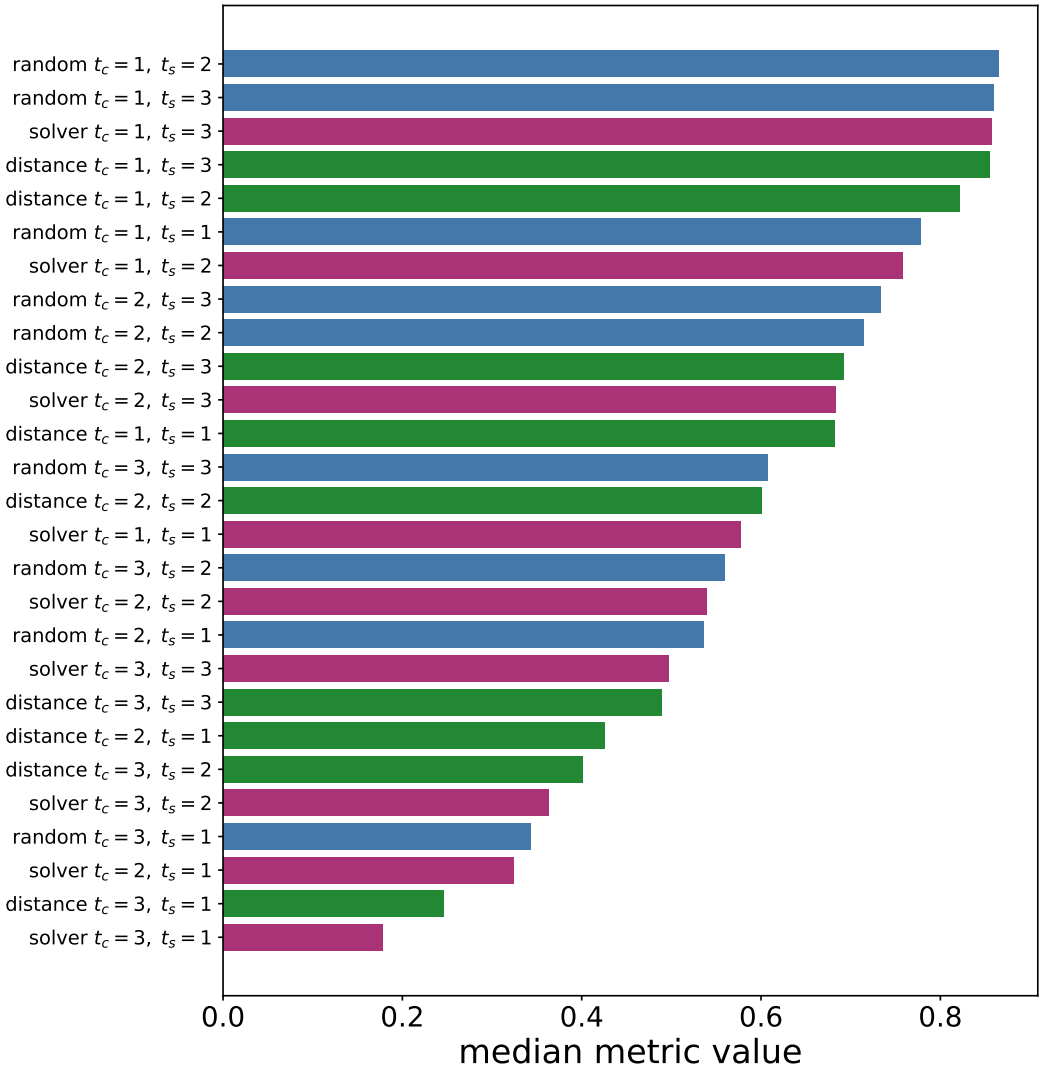


Figure A.26: Full ordering of strategy-setting combinations for MDP.

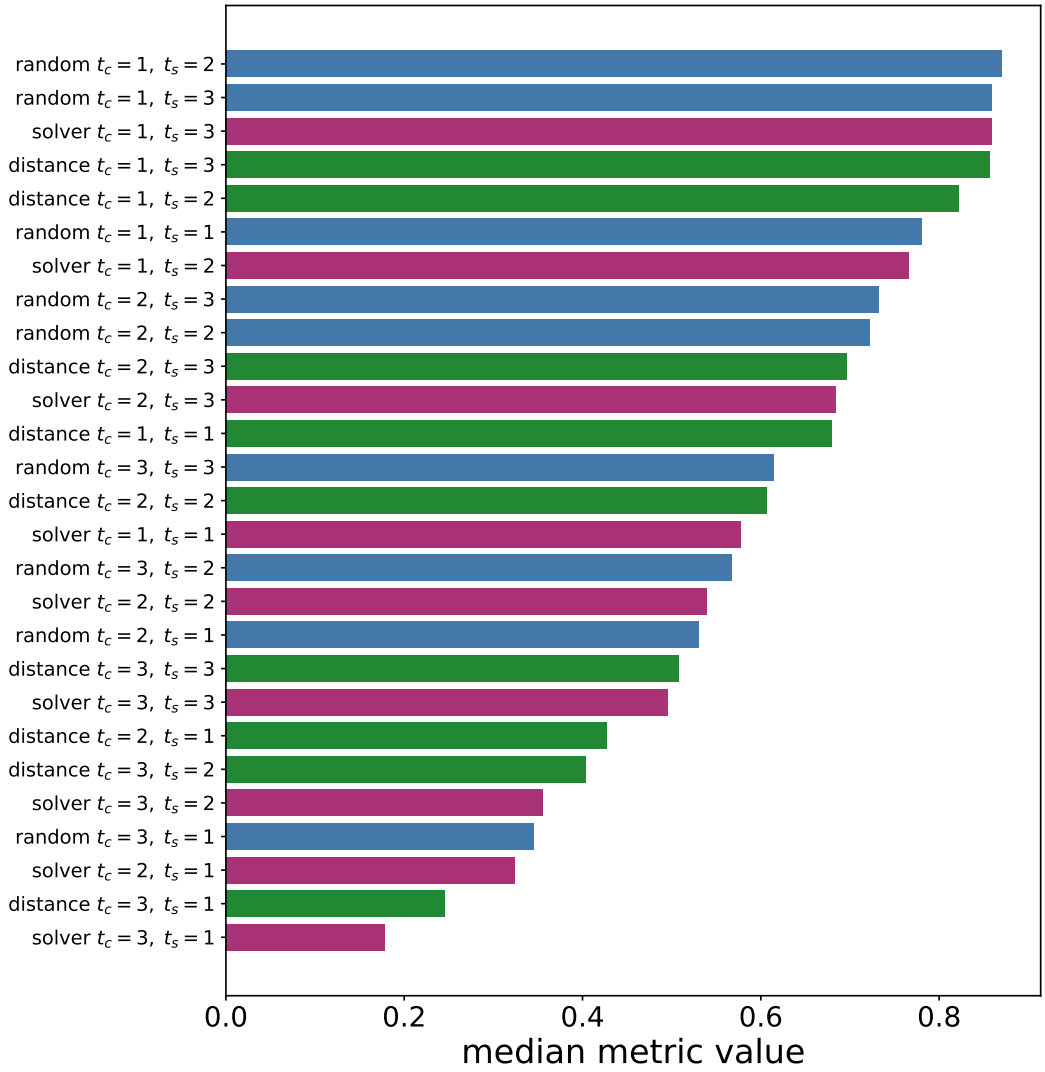


Figure A.27: Full ordering of strategy-setting combinations for MDAP.

a.3 Supplementary Results for RQ3

This section complements the results for **RQ3**. For each metric, it relates the constraint strength of a system, measured by $\log_{10}(R_i)$, to the mean achieved coverage. Each figure shows one panel per combination of coverage strength t_c and sample size t_s, with the Spearman correlation ρ and its p-value reported per panel. Systems are labelled individually, and the dashed line shows the linear trend.

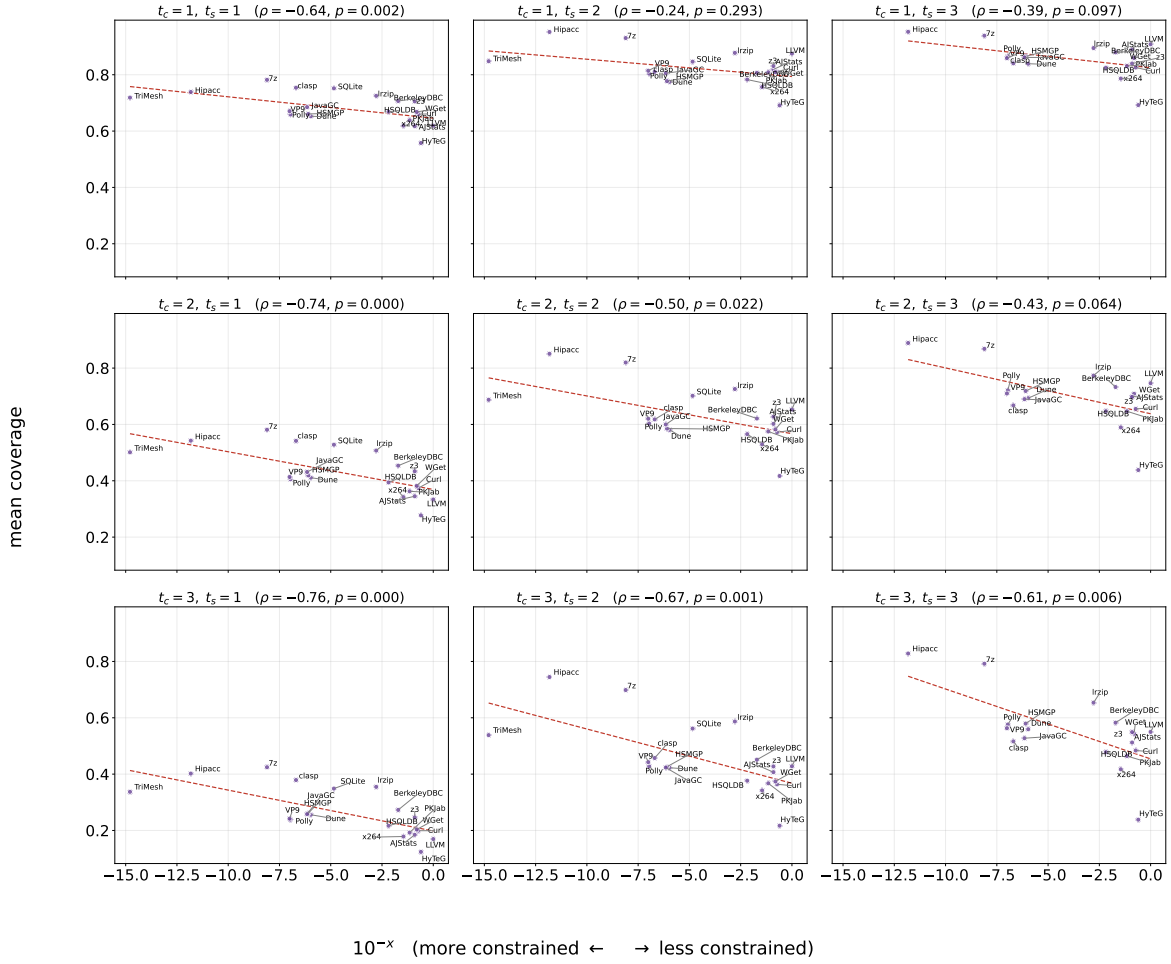


Figure A.28: Constraint strength versus mean coverage for PCI, per combination of t_c and t_s.

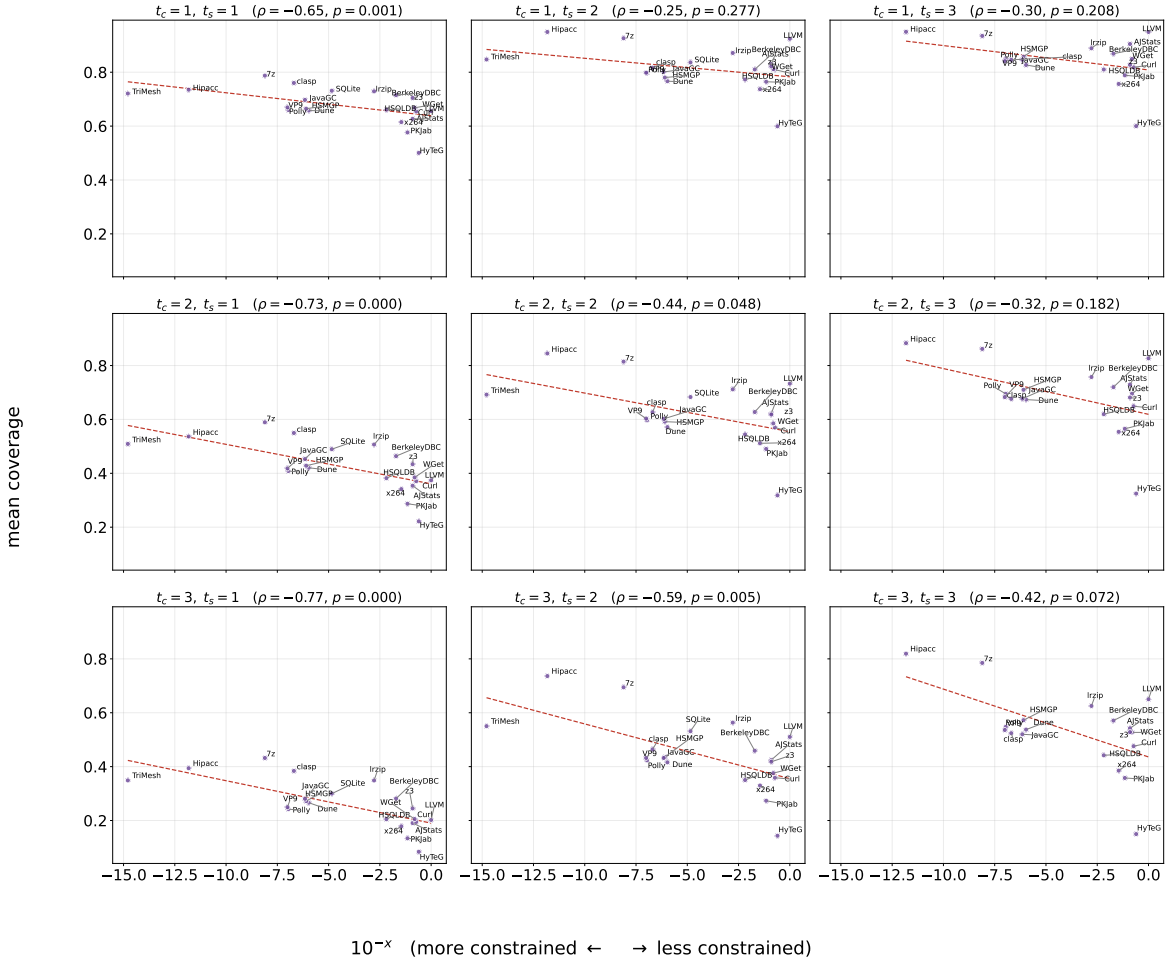


Figure A.29: Constraint strength versus mean coverage for MD, per combination of t_c and t_s.

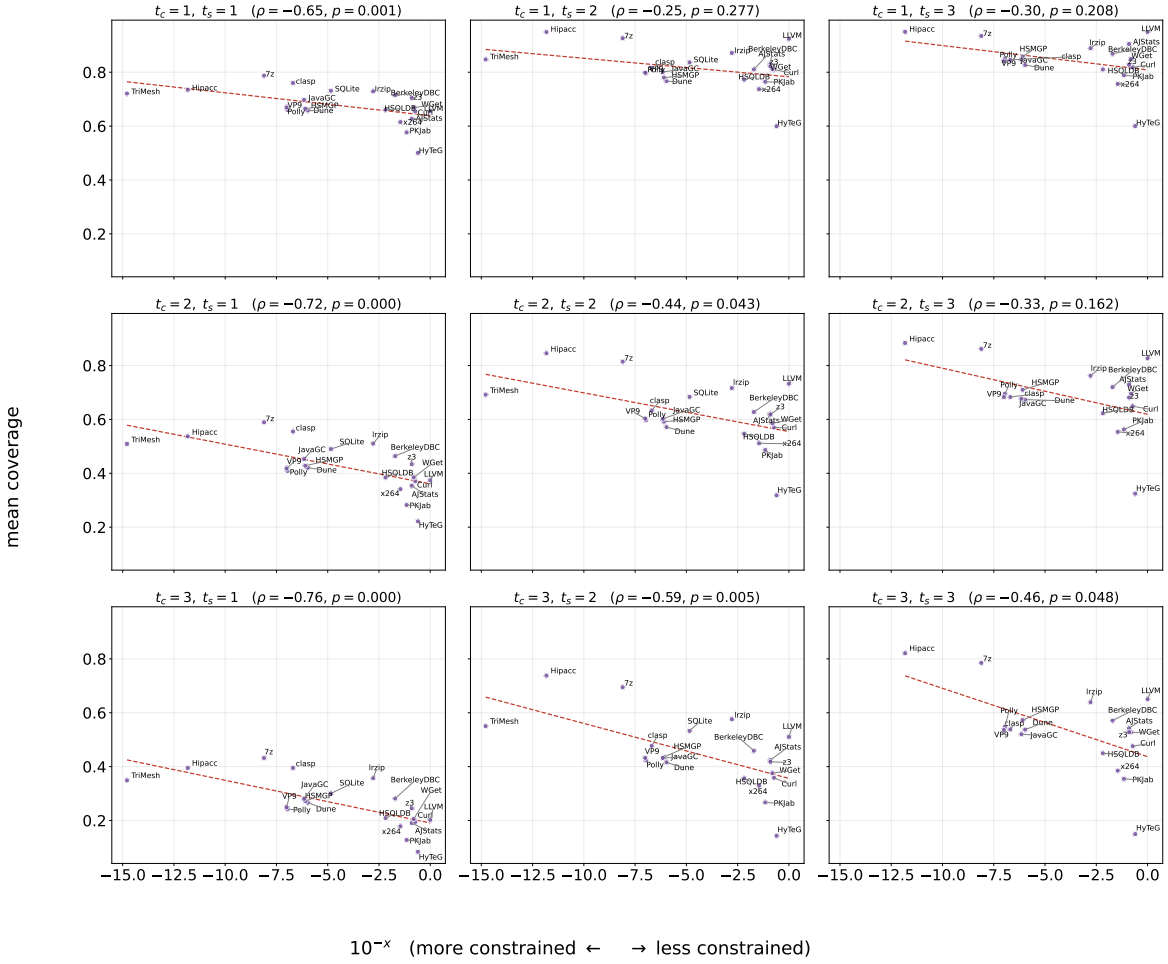


Figure A.30: Constraint strength versus mean coverage for MDP, per combination of t_c and t_s.

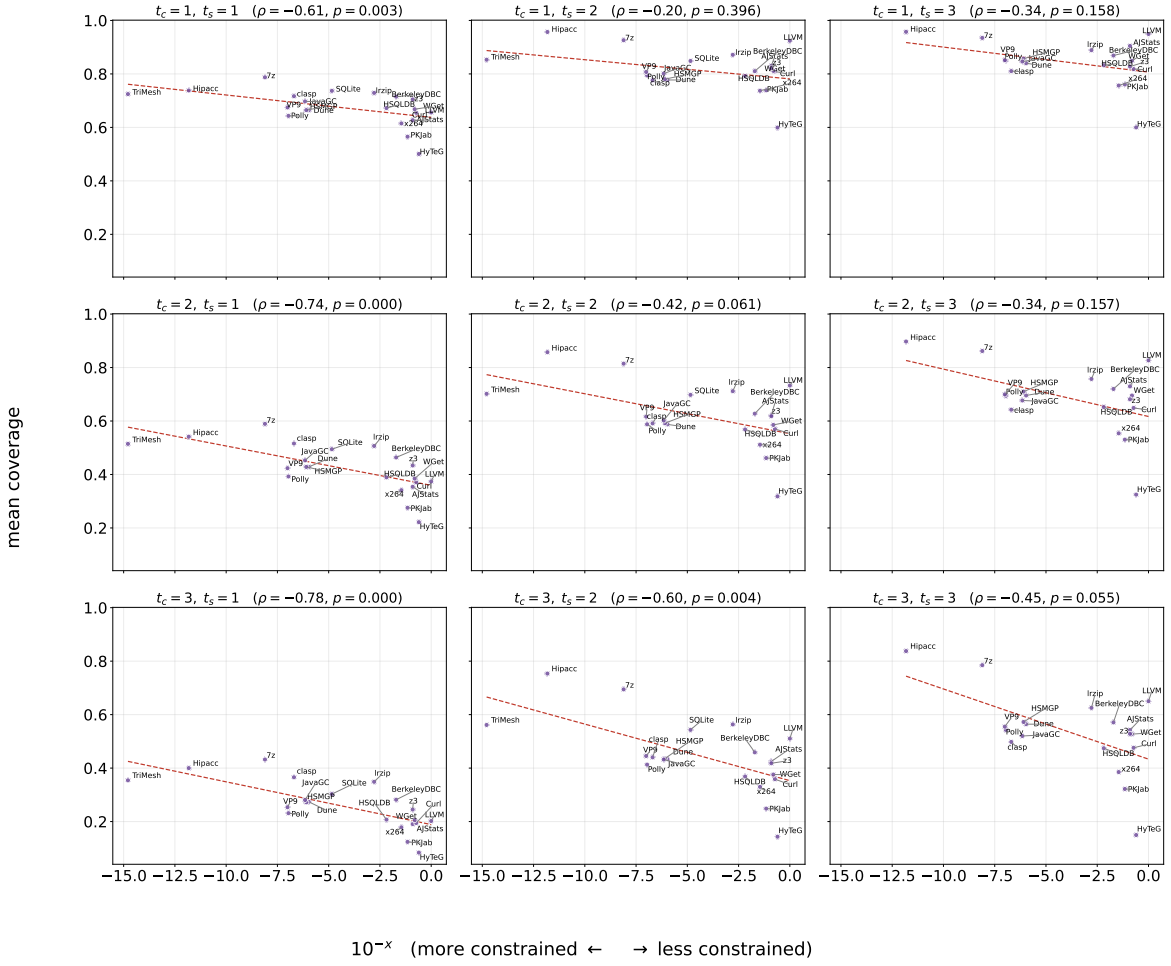


Figure A.31: Constraint strength versus mean coverage for MDA, per combination of t_c and t_s.

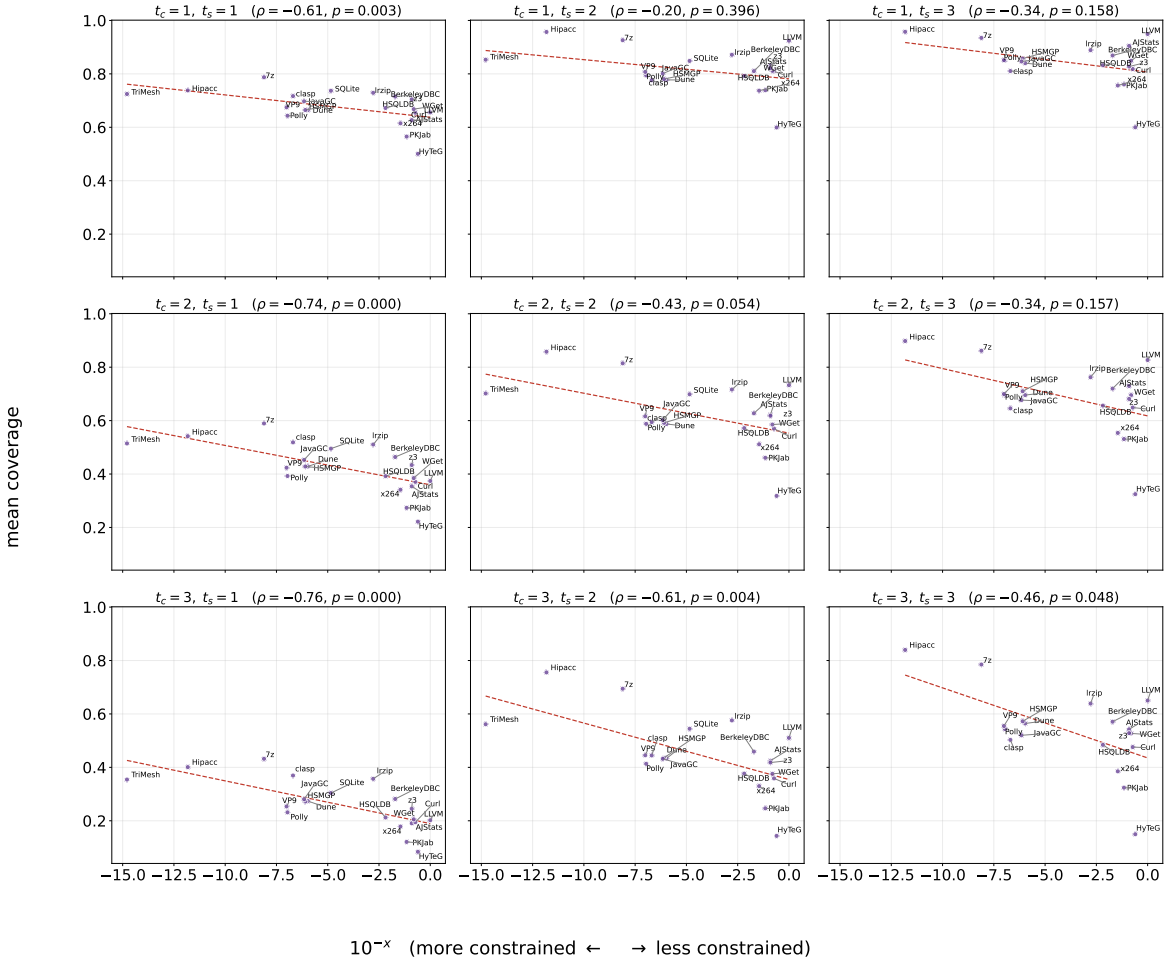


Figure A.32: Constraint strength versus mean coverage for MDAP, per combination of t_c and t_s.

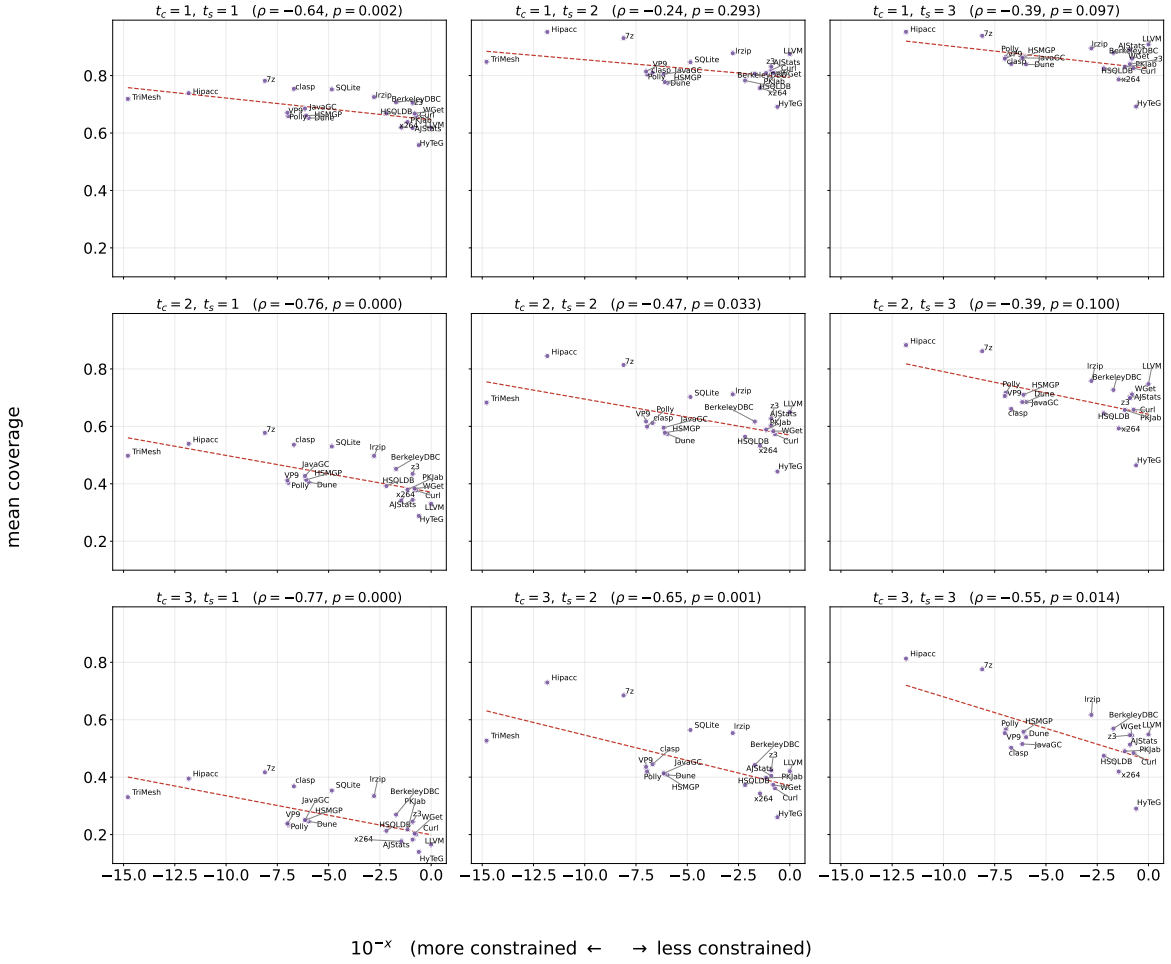


Figure A.33: Constraint strength versus mean coverage for DEFAULT, per combination of t_c and t_s.

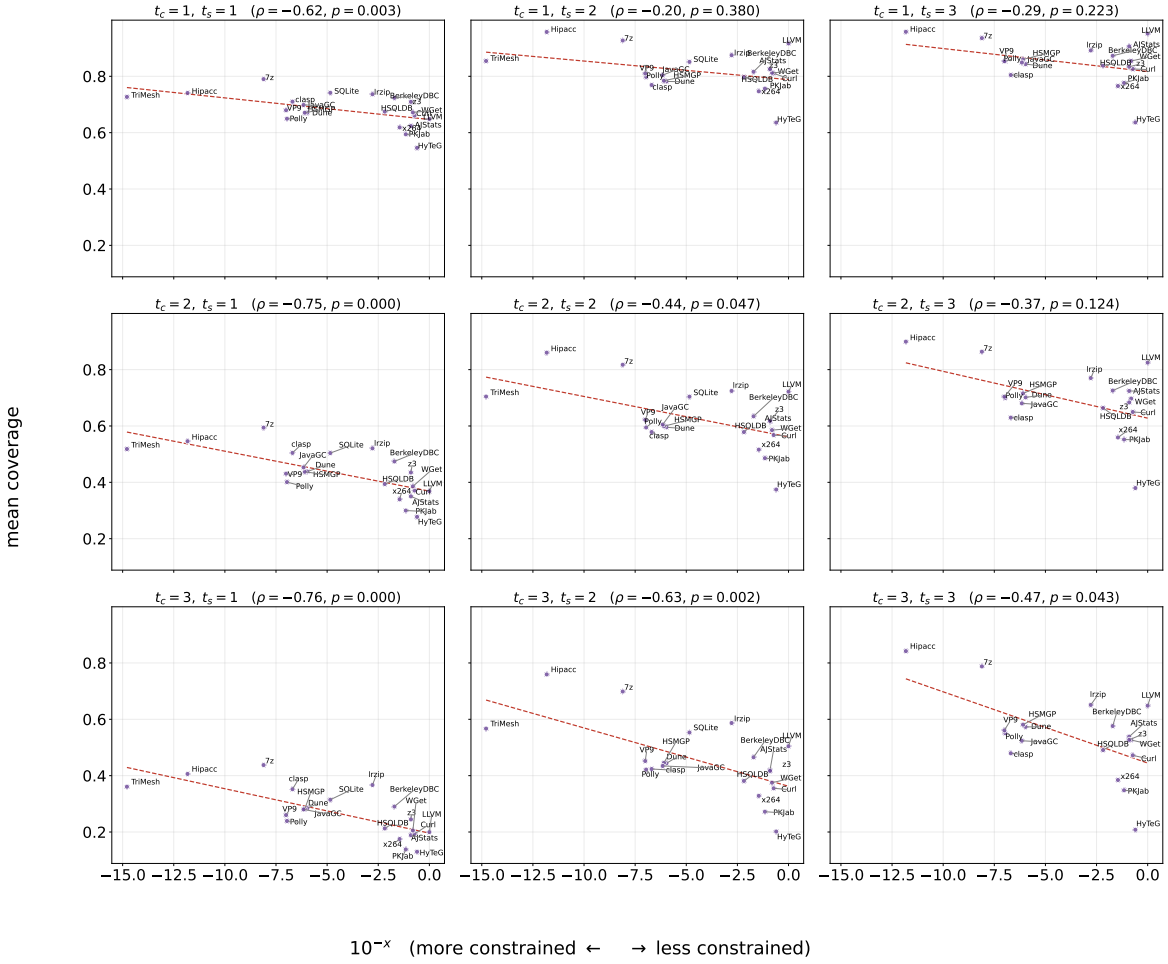


Figure A.34: Constraint strength versus mean coverage for AP, per combination of t_c and t_s.

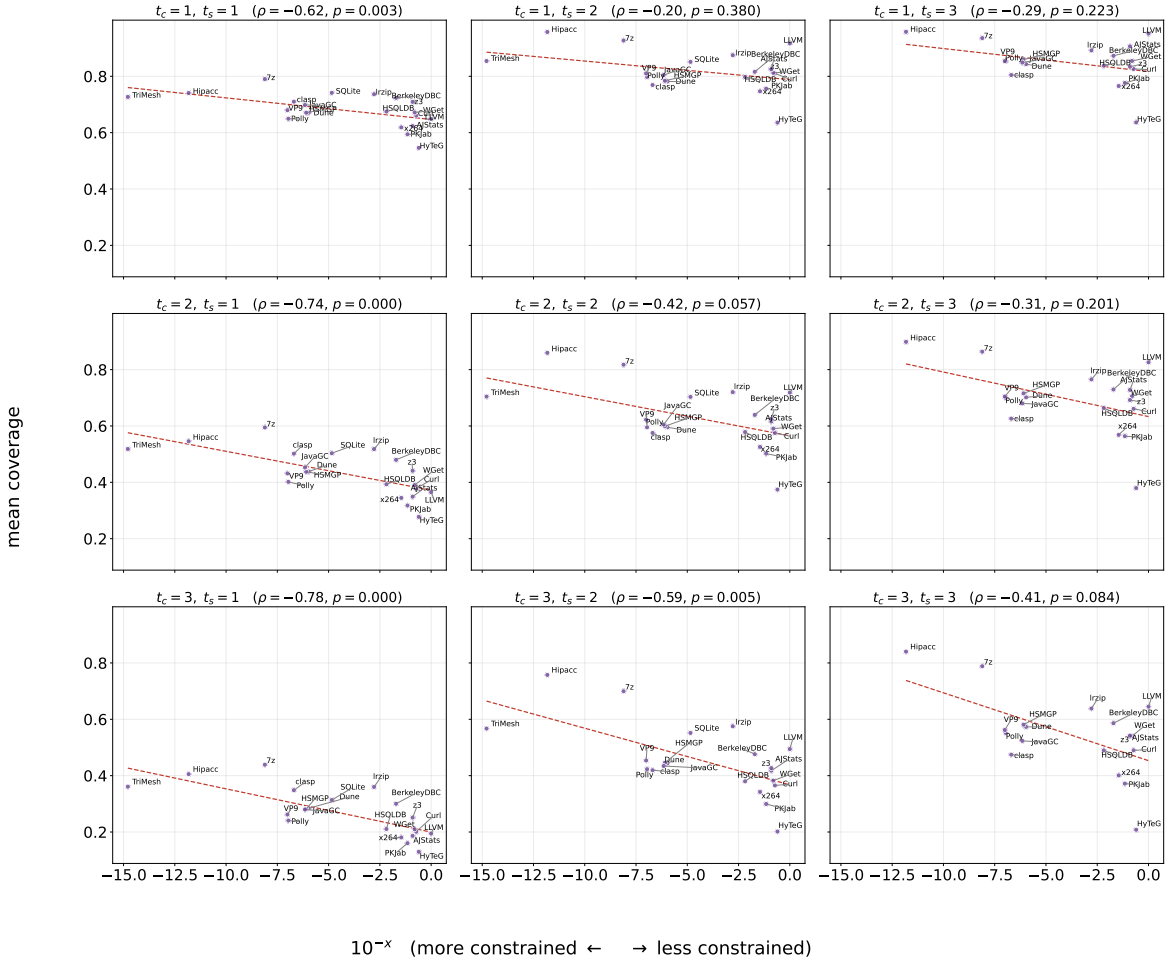


Figure A.35: Constraint strength versus mean coverage for ALS, per combination of t_c and t_s.

B

Statement on the Usage of Generative Digital Assistants

I hereby declare that I have written this thesis independently and without the involvement of third parties, and that I have used no sources or aids other than those indicated. All passages taken directly or indirectly from publications or other external sources have been identified as such. In particular, I confirm that I have used AI-based software (e.g., ChatGPT) exclusively for the following permitted sub-tasks: text rewriting/revision, check for mistakes like comma placements and spelling errors, brainstorming, code completion, and finding of bugs, and not to address or formulate the main research questions of the thesis.
I am aware of the potential risks involved in using these tools and have employed them carefully and with critical thinking. The core ideas, structure and content of this work were developed independently, without resorting to AI tools. I did not trust the results blindly, but instead checked the suggestions and only included them if they were consistent with my own understanding.

Bibliography

[1] Iago Abal, Claus Brabrand, and Andrzej Wasowski. '42 variability bugs in the linux kernel: a qualitative analysis.' In: *ACM/IEEE International Conference on Automated Software Engineering, ASE '14, Vasteras, Sweden - September 15 - 19, 2014*. Ed. by Ivica Crnkovic, Marsha Chechik, and Paul Grünbacher. ACM, 2014, pp. 421–432. DOI: 10.1145/2642937.2642990. URL: https://doi.org/10.1145/2642937.2642990.

[2] Joshua Ammermann, Tim Bittner, Domenik Eichhorn, Ina Schaefer, and Christoph Seidl. 'Can Quantum Computing Improve Uniform Random Sampling of Large Configuration Spaces? (Preprint).' In: *CoRR* abs/2307.14703 (2023). DOI: 10.48550/ARXIV.2307.14703. arXiv: 2307.14703. URL: https://doi.org/10.48550/arXiv.2307.14703.

[3] Sven Apel, Joanne M. Atlee, Luciano Baresi, and Pamela Zave. 'Feature Interactions: The Next Generation (Dagstuhl Seminar 14281).' In: *Dagstuhl Reports* 4.7 (2014), pp. 1–24. DOI: 10.4230/DAGREP.4.7.1. URL: https://doi.org/10.4230/DagRep.4.7.1.

[4] Sven Apel, Christian Lengauer, Bernhard Möller, and Christian Kästner. 'An Algebra for Features and Feature Composition.' In: *Algebraic Methodology and Software Technology, 12th International Conference, AMAST 2008, Urbana, IL, USA, July 28-31, 2008, Proceedings*. Ed. by José Meseguer and Grigore Rosu. Vol. 5140. Lecture Notes in Computer Science. Springer, 2008, pp. 36–50. DOI: 10.1007/978-3-540-79980-1_4. URL: https://doi.org/10.1007/978-3-540-79980-1_4.

[5] Sven Apel, Alexander von Rhein, Thomas Thüm, and Christian Kästner. 'Feature-interaction detection based on feature-based specifications.' In: *Comput. Networks* 57.12 (2013), pp. 2399–2409. DOI: 10.1016/J.COMNET.2013.02.025. URL: https://doi.org/10.1016/j.comnet.2013.02.025.

[6] Sven Apel, Hendrik Speidel, Philipp Wendler, Alexander von Rhein, and Dirk Beyer. 'Detection of feature interactions using feature-aware verification.' In: *26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, USA, November 6-10, 2011*. Ed. by Perry Alexander, Corina S. Pasareanu, and John G. Hosking. IEEE Computer Society, 2011, pp. 372–375. DOI: 10.1109/ASE.2011.6100075. URL: https://doi.org/10.1109/ASE.2011.6100075.

[7] Tim Bächle, Erik Hofmayer, Christoph König, Tobias Pett, and Ina Schaefer. 'Investigating the Effects of T-Wise Interaction Sampling for Vulnerability Discovery in Highly-Configurable Software Systems.' In: *Proceedings of the 29th ACM International Systems and Software Product Line Conference - Volume A, SPLC-A 2025, A Coruña, Spain, September 1-5, 2025*. Ed. by Miguel R. Luaces, Tirso V. Rodeiro, Sandra Greiner, José Ángel Galindo Duarte, Tao Yue, Kentaro Yoshimura, Laura Semini, Maxime Cordy,

Maider Azanza, Jacob Krüger, Gilles Perrouin, Sophie Fortz, Iris Groher, Daniel-Jesus Munoz, Klaus Schmid, Francisca Pérez, Jessie Galasso-Carbonnel, José Miguel Horcas, and Kevin Feichtinger. ACM, 2025, pp. 45–56. DOI: 10.1145/3744915.3748462. URL: https://doi.org/10.1145/3744915.3748462.

[8] Amitav Banerjee, Umesh B Chitnis, Sudhir L Jadhav, Jitendra Shyamsundar Bhawalkar, and Suprakash Chaudhury. 'Hypothesis testing, type I and type II errors.' In: *Industrial psychiatry journal* 18.2 (2009), pp. 127–131.

[9] Don S. Batory, Jacob Neal Sarvela, and Axel Rauschmayer. 'Scaling Step-Wise Refinement.' In: *IEEE Trans. Software Eng.* 30.6 (2004), pp. 355–371. DOI: 10.1109/TSE.2004.23. URL: https://doi.org/10.1109/TSE.2004.23.

[10] Maurice H. ter Beek, Ferruccio Damiani, Michael Lienhardt, Franco Mazzanti, and Luca Paolini. 'Static Analysis of Featured Transition Systems.' In: *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A*. SPLC '19. Paris, France: Association for Computing Machinery, 2019, 39–51. ISBN: 9781450371384. DOI: 10.1145/3336294.3336295. URL: https://doi.org/10.1145/3336294.3336295.

[11] David Benavides, Sergio Segura, and Antonio Ruiz-Cortés. 'Automated analysis of feature models 20 years later: A literature review.' In: *Information Systems* 35.6 (2010), pp. 615–636. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2010.01.001. URL: https://www.sciencedirect.com/science/article/pii/S0306437910000025.

[12] David Benavides, Chico Sundermann, Kevin Feichtinger, José A. Galindo, Rick Rabiser, and Thomas Thüm. 'UVL: Feature modelling with the Universal Variability Language.' In: *J. Syst. Softw.* 225 (2025), p. 112326. DOI: 10.1016/J.JSS.2024.112326. URL: https://doi.org/10.1016/j.jss.2024.112326.

[13] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M. Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wasowski. 'A survey of variability modeling in industrial practice.' In: *The Seventh International Workshop on Variability Modelling of Software-intensive Systems, VaMoS '13, Pisa , Italy, January 23 - 25, 2013*. Ed. by Stefania Gnesi, Philippe Collet, and Klaus Schmid. ACM, 2013, 7:1–7:8. DOI: 10.1145/2430502.2430513. URL: https://doi.org/10.1145/2430502.2430513.

[14] Daniel Le Berre and Anne Parrain. 'The Sat4j library, release 2.2.' In: *J. Satisf. Boolean Model. Comput.* 7.2-3 (2010), pp. 59–6. DOI: 10.3233/SAT190075. URL: https://doi.org/10.3233/sat190075.

[15] Sabrina Böhm, Tim Jannik Schmidt, Sebastian Krieter, Tobias Pett, Thomas Thüm, and Malte Lochau. 'Coverage Metrics for T-Wise Feature Interactions.' In: *IEEE Conference on Software Testing, Verification and Validation, ICST 2025, Napoli, Italy, March 31 - April 4, 2025*. IEEE, 2025, pp. 198–209. DOI: 10.1109/ICST62969.2025.10989003. URL: https://doi.org/10.1109/ICST62969.2025.10989003.

[16] Muffy Calder, Mario Kolberg, Evan H. Magill, and Stephan Reiff-Marganiec. 'Feature interaction: a critical review and considered forecast.' In: *Comput. Networks* 41.1 (2003), pp. 115–141. DOI: 10.1016/S1389-1286(02)00352-3. URL: https://doi.org/10.1016/S1389-1286(02)00352-3.

[17] Muffy Calder and Alice Miller. 'Feature interaction detection by pairwise analysis of LTL properties - A case study.' In: *Formal Methods Syst. Des.* 28.3 (2006), pp. 213–261. DOI: 10.1007/S10703-006-0002-5. URL: https://doi.org/10.1007/s10703-006-0002-5.

[18] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, and Moshe Y. Vardi. 'Distribution-Aware Sampling and Weighted Model Counting for SAT.' In: *CoRR* abs/1404.2984 (2014). arXiv: 1404.2984. URL: http://arxiv.org/abs/1404.2984.

[19] Andreas Classen, Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, Axel Legay, and Jean-François Raskin. 'Featured Transition Systems: Foundations for Verifying Variability Intensive Systems and Their Application to LTL Model Checking.' In: *IEEE Trans. Software Eng.* 39.8 (2013), pp. 1069–1089. DOI: 10.1109/TSE.2012.86. URL: https://doi.org/10.1109/TSE.2012.86.

[20] Duncan Cramer and Dennis Laurence Howitt. 'The Sage dictionary of statistics: a practical resource for students in the social sciences.' In: (2004).

[21] Krzysztof Czarnecki. 'Generative programming - principles and techniques of software engineering based on automated configuration and fragment-based component models.' PhD thesis. Technische Universität Illmenau, Germany, 1999. URL: https://d-nb.info/958706700.

[22] Clemens Dubslaff, Kallistos Weis, Christel Baier, and Sven Apel. 'Causality in Configurable Software Systems.' In: *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*. ACM, 2022, pp. 325–337. DOI: 10.1145/3510003.3510200. URL: https://doi.org/10.1145/3510003.3510200.

[23] Clemens Dubslaff, Kallistos Weis, Christel Baier, and Sven Apel. 'Feature causality.' In: *J. Syst. Softw.* 209 (2024), p. 111915. DOI: 10.1016/J.JSS.2023.111915. URL: https://doi.org/10.1016/j.jss.2023.111915.

[24] Olive Jean Dunn. 'Multiple comparisons using rank sums.' In: *Technometrics* 6.3 (1964), pp. 241–252.

[25] Fischer Ferreira, João Paulo Diniz, Cleiton Silva Tavares, and Eduardo Figueiredo. 'Testing Tools for Configurable Software Systems: A Review-based Empirical Study.' In: *Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems, VAMOS 2019, Leuven, Belgium, February 06-08, 2019*. Ed. by Danny Weyns and Gilles Perrouin. ACM, 2019, 6:1–6:10. DOI: 10.1145/3302333.3302344. URL: https://doi.org/10.1145/3302333.3302344.

[26] Jingzhi Gong and Tao Chen. 'Deep Configuration Performance Learning: A Systematic Survey and Taxonomy.' In: *ACM Trans. Softw. Eng. Methodol.* 34.1 (2025), 25:1–25:62. DOI: 10.1145/3702986. URL: https://doi.org/10.1145/3702986.

[27] Alexander Grebhahn, Norbert Siegmund, and Sven Apel. 'Predicting Performance of Software Configurations: There is no Silver Bullet.' In: *CoRR* abs/1911.12643 (2019). arXiv: 1911.12643. URL: http://arxiv.org/abs/1911.12643.

[28] Jianmei Guo, Krzysztof Czarnecki, Sven Apel, Norbert Siegmund, and Andrzej Wasowski. 'Variability-aware performance prediction: A statistical learning approach.' In: *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013, Silicon Valley, CA, USA, November 11-15, 2013*. Ed. by Ewen Denney, Tevfik Bultan, and Andreas Zeller. IEEE, 2013, pp. 301–311. DOI: 10.1109/ASE.2013.6693089. URL: https://doi.org/10.1109/ASE.2013.6693089.

[29] Jianmei Guo, Dingyu Yang, Norbert Siegmund, Sven Apel, Atrisha Sarkar, Pavel Valov, Krzysztof Czarnecki, Andrzej Wasowski, and Huiqun Yu. 'Data-efficient performance learning for configurable systems.' In: *Empir. Softw. Eng.* 23.3 (2018), pp. 1826–1867. DOI: 10.1007/S10664-017-9573-6. URL: https://doi.org/10.1007/s10664-017-9573-6.

[30] Joseph Y. Halpern and Judea Pearl. 'Causes and Explanations: A Structural-Model Approach: Part 1: Causes.' In: *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*. Ed. by Jack S. Breese and Daphne Koller. Morgan Kaufmann, 2001, pp. 194–202. URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1\&smnu=2\&article_id=100\&proceeding_id=17.

[31] Christopher Henard, Mike Papadakis, Mark Harman, and Yves Le Traon. 'Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines.' In: *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1*. Ed. by Antonia Bertolino, Gerardo Canfora, and Sebastian G. Elbaum. IEEE Computer Society, 2015, pp. 517–528. DOI: 10.1109/ICSE.2015.69. URL: https://doi.org/10.1109/ICSE.2015.69.

[32] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, Patrick Heymans, and Yves Le Traon. 'Bypassing the Combinatorial Explosion: Using Similarity to Generate and Prioritize T-Wise Test Configurations for Software Product Lines.' In: *IEEE Trans. Software Eng.* 40.7 (2014), pp. 650–670. DOI: 10.1109/TSE.2014.2327020. URL: https://doi.org/10.1109/TSE.2014.2327020.

[33] Ruben Heradio, David Fernández-Amorós, José A. Galindo, David Benavides, and Don S. Batory. 'Uniform and scalable sampling of highly configurable systems.' In: *Empir. Softw. Eng.* 27.2 (2022), p. 44. DOI: 10.1007/S10664-021-10102-5. URL: https://doi.org/10.1007/s10664-021-10102-5.

[34] Sture Holm. 'A simple sequentially rejective multiple test procedure.' In: *Scandinavian journal of statistics* (1979), pp. 65–70.

[35] Martin Fagereng Johansen, Øystein Haugen, and Franck Fleurey. 'An algorithm for generating t-wise covering arrays from large feature models.' In: *16th International Software Product Line Conference, SPLC '12, Salvador, Brazil - September 2-7, 2012, Volume 1*. Ed. by Eduardo Santana de Almeida, Christa Schwanninger, and David Benavides. ACM, 2012, pp. 46–55. DOI: 10.1145/2362536.2362547. URL: https://doi.org/10.1145/2362536.2362547.

[36] Nikolai Käfer, Sven Apel, Christel Baier, Clemens Dubslaff, and Holger Hermanns. 'When to Sample from Feature Diagrams?' In: *Proceedings of the 19th International Working Conference on Variability Modelling of Software-Intensive Systems, VaMoS 2025, Rennes, France, February 4-6, 2025*. Ed. by Mathieu Acher, Juliana Alves Pereira, and

Clément Quinton. ACM, 2025, pp. 11–20. DOI: 10.1145/3715340.3715442. URL: https://doi.org/10.1145/3715340.3715442.

[37] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, Jianmei Guo, and Sven Apel. 'Distance-based sampling of software configuration spaces.' In: *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*. Ed. by Joanne M. Atlee, Tevfik Bultan, and Jon Whittle. IEEE / ACM, 2019, pp. 1084–1094. DOI: 10.1109/ICSE.2019.00112. URL: https://doi.org/10.1109/ICSE.2019.00112.

[38] Kyo Kang, Sholom Cohen, James Hess, William Novak, and A. Peterson. 'Feature-Oriented Domain Analysis (FODA) feasibility study.' In: (Jan. 1990).

[39] Sergiy S. Kolesnikov. 'Feature Interactions in Configurable Software Systems.' PhD thesis. University of Passau, Germany, 2019. URL: https://opus4.kobv.de/opus4-uni-passau/frontdoor/index/index/docId/673.

[40] Matthias Kowal, Sandro Schulze, and Ina Schaefer. 'Towards efficient SPL testing by variant reduction.' In: *Proceedings of the 4th international workshop on Variability & composition, VariComp@AOSD 2013, Fukuoka, Japan, March 26, 2013*. Ed. by Walter Cazzola, Sebastián González, Dimitri Van Landuyt, and Philippe Lahire. ACM, 2013, pp. 1–6. DOI: 10.1145/2451617.2451619. URL: https://doi.org/10.1145/2451617.2451619.

[41] Sebastian Krieter, Thomas Thüm, Sandro Schulze, Sebastian Ruland, Malte Lochau, Gunter Saake, and Thomas Leich. 'T-Wise Presence Condition Coverage and Sampling for Configurable Systems.' In: *CoRR* abs/2205.15180 (2022). DOI: 10.48550/ARXIV.2205.15180. arXiv: 2205.15180. URL: https://doi.org/10.48550/arXiv.2205.15180.

[42] Dominik Michael Krupke, Ahmad Moradi, Michael Perk, Phillip Keldenich, Gabriel Gehrke, Sebastian Krieter, Thomas Thüm, and Sándor P. Fekete. 'How Low Can We Go? Minimizing Interaction Samples for Configurable Systems.' In: *ACM Trans. Softw. Eng. Methodol.* 34.6 (2025), 175:1–175:29. DOI: 10.1145/3712193. URL: https://doi.org/10.1145/3712193.

[43] William H Kruskal and W Allen Wallis. 'Use of ranks in one-criterion variance analysis.' In: *Journal of the American statistical Association* 47.260 (1952), pp. 583–621.

[44] Malte Lochau, Sebastian Oster, Ursula Goltz, and Andy Schürr. 'Model-based pairwise testing for feature interaction coverage in software product line engineering.' In: *Softw. Qual. J.* 20.3-4 (2012), pp. 567–604. DOI: 10.1007/S11219-011-9165-4. URL: https://doi.org/10.1007/s11219-011-9165-4.

[45] Malte Lochau, Ina Schaefer, Jochen Kamischke, and Sascha Lity. 'Incremental Model-Based Testing of Delta-Oriented Software Product Lines.' In: *Tests and Proofs - 6th International Conference, TAP@TOOLS 2012, Prague, Czech Republic, May 31 - June 1, 2012. Proceedings*. Ed. by Achim D. Brucker and Jacques Julliand. Vol. 7305. Lecture Notes in Computer Science. Springer, 2012, pp. 67–82. DOI: 10.1007/978-3-642-30473-6_7. URL: https://doi.org/10.1007/978-3-642-30473-6_7.

[46] Flávio Medeiros, Christian Kästner, Márcio Ribeiro, Rohit Gheyi, and Sven Apel. 'A comparison of 10 sampling algorithms for configurable systems.' In: *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*. Ed. by Laura K. Dillon, Willem Visser, and Laurie A. Williams. ACM, 2016, pp. 643–654. DOI: 10.1145/2884781.2884793. URL: https://doi.org/10.1145/2884781.2884793.

[47] Daniel-Jesus Munoz, Jeho Oh, Mónica Pinto, Lidia Fuentes, and Don S. Batory. 'Uniform random sampling product configurations of feature models that have numerical features.' In: *Proceedings of the 23rd International Systems and Software Product Line Conference, SPLC 2019, Volume A, Paris, France, September 9-13, 2019*. Ed. by Thorsten Berger, Philippe Collet, Laurence Duchien, Thomas Fogdal, Patrick Heymans, Timo Kehrer, Jabier Martinez, Raúl Mazo, Leticia Montalvillo, Camille Salinesi, Xhevahire Tërnava, Thomas Thüm, and Tewfik Ziadi. ACM, 2019, 39:1–39:13. DOI: 10.1145/3336294.3336297. URL: https://doi.org/10.1145/3336294.3336297.

[48] KimHao Nguyen and ThanhVu Nguyen. 'GenTree: Using Decision Trees to Learn Interactions for Configurable Software.' In: *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 2021, pp. 1598–1609. DOI: 10.1109/ICSE43902.2021.00142. URL: https://doi.org/10.1109/ICSE43902.2021.00142.

[49] ThanhVu Nguyen, Ugur Koc, Javran Cheng, Jeffrey S. Foster, and Adam A. Porter. 'iGen: Dynamic Interaction Inference for Configurable Software.' In: *CoRR* abs/1903.12247 (2019). arXiv: 1903.12247. URL: http://arxiv.org/abs/1903.12247.

[50] Jeho Oh, Paul Gazzillo, and Don S. Batory. '*t*-wise coverage by uniform sampling.' In: *Proceedings of the 23rd International Systems and Software Product Line Conference, SPLC 2019, Volume A, Paris, France, September 9-13, 2019*. Ed. by Thorsten Berger, Philippe Collet, Laurence Duchien, Thomas Fogdal, Patrick Heymans, Timo Kehrer, Jabier Martinez, Raúl Mazo, Leticia Montalvillo, Camille Salinesi, Xhevahire Tërnava, Thomas Thüm, and Tewfik Ziadi. ACM, 2019, 15:1–15:4. DOI: 10.1145/3336294.3342359. URL: https://doi.org/10.1145/3336294.3342359.

[51] R Lyman Ott. *An introduction to statistical methods and data analysis*. 2001.

[52] Gilles Perrouin, Sebastian Oster, Sagar Sen, Jacques Klein, Benoit Baudry, and Yves Le Traon. 'Pairwise testing for software product lines: comparison of two approaches.' In: *Softw. Qual. J.* 20.3-4 (2012), pp. 605–643. DOI: 10.1007/S11219-011-9160-9. URL: https://doi.org/10.1007/s11219-011-9160-9.

[53] Tobias Pett, Sebastian Krieter, Thomas Thüm, and Ina Schaefer. 'MulTi-Wise Sampling: Trading Uniform T-Wise Feature Interaction Coverage for Smaller Samples.' In: *CoRR* abs/2406.19801 (2024). DOI: 10.48550/ARXIV.2406.19801. arXiv: 2406.19801. URL: https://doi.org/10.48550/arXiv.2406.19801.

[54] Quentin Plazar, Mathieu Acher, Gilles Perrouin, Xavier Devroey, and Maxime Cordy. 'Uniform Sampling of SAT Solutions for Configurable Systems: Are We There Yet?' In: *12th IEEE Conference on Software Testing, Validation and Verification, ICST 2019, Xi'an, China, April 22-27, 2019*. IEEE, 2019, pp. 240–251. DOI: 10.1109/ICST.2019.00032. URL: https://doi.org/10.1109/ICST.2019.00032.

[55] Hendrik Post and Carsten Sinz. 'Configuration Lifting: Verification meets Software Configuration.' In: *23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), 15-19 September 2008, L'Aquila, Italy*. IEEE Computer Society, 2008, pp. 347–350. DOI: 10.1109/ASE.2008.45. URL: https://doi.org/10.1109/ASE.2008.45.

[56] Alexander von Rhein, Jörg Liebig, Andreas Janker, Christian Kästner, and Sven Apel. 'Variability-Aware Static Analysis at Scale: An Empirical Study.' In: *ACM Trans. Softw. Eng. Methodol.* 27.4 (2018), 18:1–18:33. DOI: 10.1145/3280986. URL: https://doi.org/10.1145/3280986.

[57] Florian Sattler. 'Understanding variability in space and time: analyzing features and revisions in concert.' PhD thesis. Saarland University, Saarbrücken, Germany, 2023. URL: https://publikationen.sulb.uni-saarland.de/handle/20.500.11880/37623.

[58] Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. 'Performance-influence models for highly configurable systems.' In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015*. Ed. by Elisabetta Di Nitto, Mark Harman, and Patrick Heymans. ACM, 2015, pp. 284–294. DOI: 10.1145/2786805.2786845. URL: https://doi.org/10.1145/2786805.2786845.

[59] Norbert Siegmund, Sergiy S. Kolesnikov, Christian Kästner, Sven Apel, Don S. Batory, Marko Rosenmüller, and Gunter Saake. 'Predicting performance via automated feature-interaction detection.' In: *34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland*. Ed. by Martin Glinz, Gail C. Murphy, and Mauro Pezzè. IEEE Computer Society, 2012, pp. 167–177. DOI: 10.1109/ICSE.2012.6227196. URL: https://doi.org/10.1109/ICSE.2012.6227196.

[60] Larissa Rocha Soares. 'VarXplorer: reasoning about feature interactions.' In: *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*. Ed. by Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman. ACM, 2018, pp. 500–502. DOI: 10.1145/3183440.3190329. URL: https://doi.org/10.1145/3183440.3190329.

[61] C. Spearman. 'The Proof and Measurement of Association between Two Things.' In: *The American Journal of Psychology* 100.3/4 (1987), pp. 441–471. ISSN: 00029556. URL: http://www.jstor.org/stable/1422689.

[62] Thomas Thüm, Sven Apel, Christian Kästner, Ina Schaefer, and Gunter Saake. 'A Classification and Survey of Analysis Strategies for Software Product Lines.' In: *ACM Comput. Surv.* 47.1 (June 2014). ISSN: 0360-0300. DOI: 10.1145/2580950. URL: https://doi.org/10.1145/2580950.

[63] Engin Uzuncaova, Sarfraz Khurshid, and Don S. Batory. 'Incremental Test Generation for Software Product Lines.' In: *IEEE Trans. Software Eng.* 36.3 (2010), pp. 309–322. DOI: 10.1109/TSE.2010.30. URL: https://doi.org/10.1109/TSE.2010.30.

[64] Mahsa Varshosaz, Mustafa Al-Hajjaji, Thomas Thüm, Tobias Runge, Mohammad Reza Mousavi, and Ina Schaefer. 'A classification of product sampling for software product lines.' In: *Proceedings of the 22nd International Systems and Software Product Line Conference - Volume 1*. SPLC '18. Gothenburg, Sweden: Association for Computing

Machinery, 2018, 1–13. ISBN: 9781450364645. DOI: 10.1145/3233027.3233035. URL: https://doi.org/10.1145/3233027.3233035.

[65] Miguel Velez, Pooyan Jamshidi, Florian Sattler, Norbert Siegmund, Sven Apel, and Christian Kästner. 'ConfigCrusher: towards white-box performance analysis for configurable systems.' In: *Autom. Softw. Eng.* 27.3 (2020), pp. 265–300. DOI: 10.1007/S10515-020-00273-8. URL: https://doi.org/10.1007/s10515-020-00273-8.

[66] Miguel Velez, Pooyan Jamshidi, Norbert Siegmund, Sven Apel, and Christian Kästner. 'White-Box Analysis over Machine Learning: Modeling Performance of Configurable Systems.' In: *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 2021, pp. 1072–1084. DOI: 10.1109/ICSE43902.2021.00100. URL: https://doi.org/10.1109/ICSE43902.2021.00100.

[67] Max Weber, Christian Kaltenecker, Florian Sattler, Sven Apel, and Norbert Siegmund. 'Twins or False Friends? A Study on Energy Consumption and Performance of Configurable Software.' In: *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*. IEEE, 2023, pp. 2098–2110. DOI: 10.1109/ICSE48619.2023.00177. URL: https://doi.org/10.1109/ICSE48619.2023.00177.

[68] Alan W. Williams and Robert L. Probert. 'A Measure for Component Interaction Test Coverage.' In: *2001 ACS / IEEE International Conference on Computer Systems and Applications (AICCSA 2001), 26-29 June 2001, Beirut, Lebanon*. IEEE Computer Society, 2001, pp. 304–312. DOI: 10.1109/AICCSA.2001.934001. URL: https://doi.org/10.1109/AICCSA.2001.934001.

[69] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering, Second Edition*. Springer, 2024. ISBN: 978-3-662-69305-6. DOI: 10.1007/978-3-662-69306-3. URL: https://doi.org/10.1007/978-3-662-69306-3.

[70] Tianyin Xu, Long Jin, Xuepeng Fan, Yuanyuan Zhou, Shankar Pasupathy, and Rukma Talwadker. 'Hey, you have given me too many knobs!: understanding and dealing with over-designed configuration in system software.' In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015*. Ed. by Elisabetta Di Nitto, Mark Harman, and Patrick Heymans. ACM, 2015, pp. 307–319. DOI: 10.1145/2786805.2786852. URL: https://doi.org/10.1145/2786805.2786852.