Bachelor's Thesis

Understanding the Impact of Sampling on Subgroup Discovery in Configuration Spaces

Marcel Varga

February 1, 2026

Advisor:
Tobias Dick Chair of Software Engineering

Examiners:
Prof. Dr. Sven Apel Chair of Software Engineering
Prof. Dr. Jilles Vreeken CISPA Helmholtz Center for Information Security

Chair of Software Engineering
Saarland Informatics Campus
Saarland University

UNIVERSITÄT
DES
SAARLANDES

Erklärung Statement

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 zur Textüberarbeitung/-korrektur und zur Code-Vervollständigung und nicht zur Bearbeitung der in der Arbeit aufgeworfenen Fragestellungen zu Hilfe genommen habe. Alle mittels künstlicher Intelligenz betriebenen Software (z. B. ChatGPT) generierten und/oder bearbeiteten Teile der Arbeit wurden kenntlich gemacht und als Hilfsmittel angegeben. Ich erkläre mich damit einverstanden, dass die Arbeit mittels eines Plagiatsprogrammes auf die Nutzung einer solchen Software überprüft wird. Mir ist bewusst, dass der Verstoß gegen diese Versicherung zum Nichtbestehen der Prüfung bis hin zum Verlust des Prüfungsanspruchs führen kann.

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 and code completion, and not to address or formulate the main research questions of the thesis. All parts of the thesis that were generated and/or edited using AI-based software (e.g., ChatGPT) have been disclosed and documented in accordance with the documentation requirements. I agree that the thesis may be checked using plagiarism detection software, including checks for the use of such software. I am aware that any violation of this declaration may result in failing the examination and lead to losing the right to be examined.

Saarbrücken, _____ _____
(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 agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, _____ _____
(Datum Date) (Unterschrift Signature)

Abstract

Configurable software systems expose large configuration spaces, which makes exhaustive performance analysis infeasible in practice. To reduce measurement cost, sampling strategies are commonly used to select a representative subset of configurations for performance evaluation. This subset can then be analysed using performance models or data mining methods, such as subgroup discovery, to identify notable configuration patterns. However, subgroup discovery methods rely on both structural patterns and performance distributions, which may be distorted by sampling. It is therefore unclear how well different sampling strategies and sample sizes preserve the information required for reliable subgroup discovery.

In this thesis, we conduct an empirical study on eight real-world configurable software systems to investigate how sampling strategy and sample size influence subgroup discovery outcomes on the example of SYFLOW. We compare subgroups discovered from sampled and full datasets to analyse how well sampling preserves subgroup characteristics. Our findings indicate that sampling strategies primarily affect the stability of subgroup structure rather than the preservation of performance distributions. Moreover, they suggest that sample sizes of around 20% of the configuration space already capture most of the attainable reliability for subgroup discovery, with larger samples yielding diminishing returns.

Contents

List of Figures

List of Tables

1 Introduction

Modern software systems expose a large number of configuration options to tailor behaviour and performance to specific contexts [33]. This flexibility allows developers and users to adapt systems to different hardware platforms, workloads, or quality constraints. Examples include database systems tuned for latency or throughput, compilers and video encoders with numerous optimisation flags, operating systems such as *Linux*, and configurable cloud services with complex performance trade-offs [1, 20, 28]. While this configurability is essential for achieving good performance across diverse scenarios, it also gives rise to highly complex configuration spaces whose size often grows exponentially with the number of options [1].

Moreover, interactions between configuration options can introduce unexpected effects on runtime, throughput, or latency, such that relatively small regions of the configuration space may exhibit behaviour that differs substantially from the overall population. Identifying and characterising these exceptional regions is therefore a central challenge in performance analysis, as it supports debugging, performance tuning, and system understanding [10, 20, 28]. However, exhaustive measurement of all valid configurations is often infeasible in practice due to time and resource constraints [19, 26].

As a consequence, *sampling strategies* are commonly employed to select a manageable subset of configurations for performance measurement. These strategies differ substantially in the aspects of the configuration space they attempt to preserve [30]. Random sampling is simple but may be unstable under constraints, while solver-based approaches guarantee validity but can introduce biases through internal heuristics [5, 6]. Combinatorial t-wise sampling ensures coverage of interactions up to order t, but becomes increasingly expensive as t grows [24]. Distance-based or diversity-oriented methods aim to explore heterogeneous regions of the configuration space [19].

Inherently, these approaches trade off different notions of representativeness, such as coverage, diversity, and uniformity. By reducing the number of measured configurations, sampling limits the available evidence for analysis and may distort both the structural composition of the configuration space and its associated performance distribution. Nevertheless, sampling is indispensable in practice, as exhaustive measurement is often too expensive or infeasible for large configuration spaces. While sampling strategies have been studied extensively in the context of fault detection [21, 26] and performance prediction [18, 28], their impact on *descriptive* performance analysis remains less well understood.

One such method for descriptive performance analysis is *subgroup discovery*, which provides a descriptive, model-agnostic approach to analysing performance data [3]. Rather than predicting performance, subgroup discovery focuses on identifying human-interpretable rules over configuration options. These rules describe subsets of configurations whose

performance deviates from that of the overall population and can reveal actionable insights into performance anomalies by highlighting specific feature combinations associated with exceptional performance behaviour. Recent work in the area of subgroup discovery includes SYFLOW, a method that formulates rule learning as a continuous optimisation problem using machine learning [32]. By avoiding explicit combinatorial enumeration and modelling performance distributions directly, SYFLOW scales to large datasets while maintaining interpretability, making it well suited for analysing real-world configurable systems.

Despite these advances, it remains unclear how different sampling strategies affect the subgroups discovered by modern methods such as SYFLOW. In particular, sampling may cause certain subgroups to disappear, shift in structure, or become unstable across repeated runs. In addition, sampling may distort performance distributions in ways that influence subgroup discovery outcomes, even when subgroup structure is preserved. Furthermore, the role of *sample size* in stabilising subgroup discovery and mitigating sampling-induced effects has not been systematically studied in this context.

To close this gap, we investigate how sampling strategies and sample size influence subgroup discovery results for performance analysis of configurable software systems. We conduct a large-scale study on eight real-world configurable systems. For each system, we generate sampled datasets using multiple established sampling strategies across different relative sampling budgets and, where applicable, multiple random seeds.

We apply SYFLOW to both full and sampled datasets to study how sampling strategies and sample sizes affect the structure and performance behaviour of discovered subgroups.

Our results show that subgroup discovery outcomes can vary substantially depending on the chosen sampling strategy and sample size. In particular, some sampling strategies preserve subgroup structure and performance behaviour more consistently than others, while increasing sample size exhibits diminishing returns beyond a certain point. These findings provide practical guidance for selecting sampling approaches when applying descriptive performance analysis to highly configurable software systems.

2

Background

This chapter introduces the fundamental concepts relevant to this thesis. In particular, it introduces key concepts from performance analysis and sampling strategies in the context of highly configurable software systems and explains how these concepts are used to identify configuration-related performance anomalies using subgroup discovery algorithms.

2.1 Configurable Software Systems

Modern software systems are often implemented with built-in configurability. Such *configurable software systems* expose a set of configuration options or *features* that change the system's behaviour. Let $F = \{f_1, f_2, \ldots, f_n\}$ denote the set of features. A concrete configuration c is an assignment of values to each feature in F. Typically, these features are implemented as boolean flags that describe if the respective feature is enabled or parameters with numeric or categorical domains altering the behaviour of the system. Consequently, the number of possible configurations grows combinatorially with the number of features.

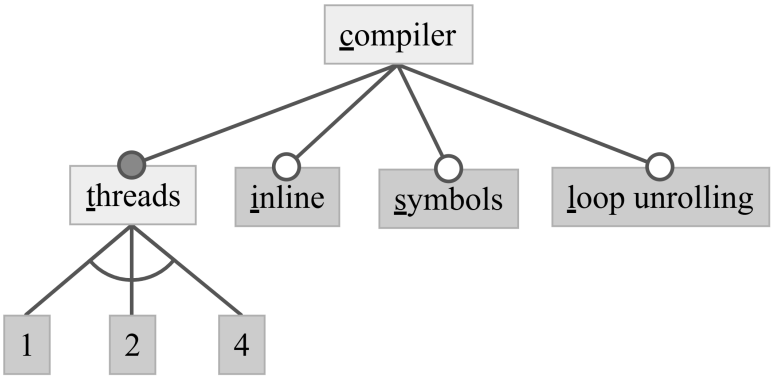

Figure 2.1: A feature model representing a compiler configuration

For example, consider the feature model presented in Figure 2.1, which describes a very basic compiler configuration. The model only includes 6 features, yet even such a small example already results in a theoretical configuration space of $2^8 = 256$ possible configurations.

However, not every assignment is valid. Domain-specific constraints, such as mutual exclusions, dependency relations, mandatory options or numeric bounds restrict the set of valid configurations. A feature model is a commonly used, declarative representation of

these constraints and can be encoded as propositional logic or a graphical diagram. They distinguish between mandatory and optional features and support group relations such as *OR* and *XOR*, which require at least or exactly one option in their respective group to be enabled, and cross-feature constraints, restricting arbitrary features to only be enabled if another is in a predetermined enabled state [1]. The set of all valid configurations, the *configuration space*, is denoted by $C \subseteq \mathcal{A}(F)$, where $\mathcal{A}(F)$ is the set of all possible assignments over F.

Returning to our example, although the total potential configuration space contains 256 configurations, structural constraints of the feature model restrict the set of valid combinations. In this example, features `c` and `t` are mandatory and must always be selected, while `i`, `l`, and `s` are optional and may be enabled independently. Additionally, features `1`, `2`, and `4` form an *XOR* group, meaning that exactly one of them must be chosen in any valid configuration. Together, these constraints reduce the valid configuration space C to the following 24 configurations:

$$\{c,t,1\},\{c,t,1,i\},\{c,t,1,s\},\{c,t,1,l\},\{c,t,1,i,s\},\{c,t,1,i,l\}$$
$$\{c,t,1,s,l\},\{c,t,1,i,s,l\},\{c,t,2\},\{c,t,2,i\},\{c,t,2,s\},\{c,t,2,l\}$$
$$\{c,t,2,i,s\},\{c,t,2,i,l\},\{c,t,2,s,l\},\{c,t,2,i,s,l\},\{c,t,4\},\{c,t,4,i\}$$
$$\{c,t,4,s\},\{c,t,4,l\},\{c,t,4,i,s\},\{c,t,4,i,l\},\{c,t,4,s,l\},\{c,t,4,i,s,l\}$$

2.2 Sampling Strategies

The cardinality $|C|$ typically grows combinatorially with $|F|$ even when constraints eliminate many combinations. As a result, exhaustive measurement of every $c \in C$ is often impractical in real-world systems. At the same time, interactions between features may cause unexpected effects. A small subset of configurations can have notable effects on performance or other non-functional properties [18].

To make analysis more manageable, *sampling strategies* are commonly employed to select a subset $\widehat{C} \subset C$ for measurement. Different strategies emphasize different trade-offs between validity, diversity, interaction coverage, and computational cost.

Random sampling aims to select configurations uniformly random from C. Typically, naive random sampling is implemented as *rejection sampling*, where samples are randomly generated over the entire assignment space and those that violate feature-model constraints are discarded. However, this approach can be inefficient and achieving uniformity becomes increasingly difficult, especially when the configuration space is highly constrained and drawing invalid configurations becomes more and more likely.

Solver-based sampling addresses the issues occurring in random sampling by using SAT solvers to generate only valid assignments. The downside of this approach is that these solvers on their own often introduce bias towards certain regions of the configuration space due to their internal heuristics and search strategies being designed for efficient problem

solving rather than uniform sampling, thus modern approaches attempt to avoid such clustering by introducing further processing for near-uniform results [5, 6].

t-wise sampling selects configurations such that for every subset of t distinct features, all valid combinations of their values occur in at least one sampled configuration [24]. This ensures that interactions among up to t features are represented in the sample. Pairwise sampling ($t = 2$) is the most common instance, offering a practical balance between interaction coverage and sampling cost. For example, in the example presented above (Figure 2.1), a pairwise sample could be:

$$\widehat{C} = \Big\{\{c, t, 1\},\ \{c, t, 2, i\},\ \{c, t, 4, s\},\ \{c, t, 1, i, s, l\}\Big\}.$$

However, even low-order t-wise sampling can be costly for large systems with many features, such as the Linux kernel. Moreover, some interactions may involve more than two features, and while pairwise sampling is often effective at exposing local interactions, it is unreliable at detecting global ones [26]. Thus, higher order t-wise sampling (e.g., $t = 3$ or $t = 4$) can capture more complex interactions, but quickly becomes infeasible as t grows.

Distance-based sampling defines a distance metric over configurations, for instance Hamming distance on boolean features or a mixed metric for mixed domains, and selects configurations to maximize diversity in that metric space, spreading samples across the configuration landscape [19]. Variants of distance-based sampling add diversification steps to ensure underrepresented feature values appear more often [19].

Each of these strategies therefore has different implications for subsequent analyses: Uniform or near-uniform samples may better preserve global statistics, t-wise sampling excels at exposing local interactions up to order t, solver-based strategies guarantee validity but risk regional bias, and distance-based strategies emphasize coverage of heterogeneity. Further, we distinguish between *stochastic* and *deterministic* sampling strategies. Stochastic strategies include random sampling, distance-based sampling, and its diversified variant, whereas deterministic strategies comprise combinatorial t-wise sampling and our implementation of solver-based sampling. Stochastic strategies rely on randomness when selecting configurations and therefore typically require multiple runs to obtain stable and reliable results. In contrast, deterministic strategies produce identical sample sets across runs for a given configuration space and sampling budget.

Note that solver-based sampling is typically implemented as a stochastic strategy. However, our implementation yields deterministic results for all datasets and sample sizes considered in this study. We therefore group solver-based sampling together with combinatorial t-wise sampling for the purpose of our evaluation.

2.3 Configuration Subspaces

We define *performance* as a surjective mapping P, such that every configuration $c \in C$ has an associated real-valued measurement $P(c) \in \mathbb{R}$, taken under a standardized workload. A *configuration subspace* is any subset $S \subseteq C$ and is typically described by a rule or predicate

over features. Returning to the compiler configuration example in Figure 2.1, consider the subspace $S = \{c \in C \mid \mathrm{i} \in c\}$, which contains all valid configurations where the *inlining (i)* feature is selected. For this example, S consists of the following 12 configurations:

$$\begin{array}{llll} \{\mathrm{c},\mathrm{t},1,\mathrm{i}\}, & \{\mathrm{c},\mathrm{t},1,\mathrm{i},\mathrm{s}\}, & \{\mathrm{c},\mathrm{t},1,\mathrm{i},\mathrm{l}\}, & \{\mathrm{c},\mathrm{t},1,\mathrm{i},\mathrm{s},\mathrm{l}\} \\ \{\mathrm{c},\mathrm{t},2,\mathrm{i}\}, & \{\mathrm{c},\mathrm{t},2,\mathrm{i},\mathrm{s}\}, & \{\mathrm{c},\mathrm{t},2,\mathrm{i},\mathrm{l}\}, & \{\mathrm{c},\mathrm{t},2,\mathrm{i},\mathrm{s},\mathrm{l}\} \\ \{\mathrm{c},\mathrm{t},4,\mathrm{i}\}, & \{\mathrm{c},\mathrm{t},4,\mathrm{i},\mathrm{s}\}, & \{\mathrm{c},\mathrm{t},4,\mathrm{i},\mathrm{l}\}, & \{\mathrm{c},\mathrm{t},4,\mathrm{i},\mathrm{s},\mathrm{l}\} \end{array}$$

A subspace S is of interest if the distribution of P over S differs substantially from its distribution over C. In this thesis, we use the term *exceptional subgroup* to refer to such a subspace whose behaviour is significantly different with respect to runtime performance.

2.4 Subgroup Discovery

Subgroup discovery is a descriptive, local-pattern mining technique whose goal is to find interpretable descriptions of subsets (subgroups) of a dataset that show a notable behaviour with respect to a predefined property of interest. In the context of configurable software systems, it can be used to find configuration subspaces where a certain non-functional property, in our case runtime performance, deviates significantly from the overall system behaviour [3, 23].

Typically, subgroup discovery uses a quality function $Q : 2^F \mapsto \mathbb{R}$, and a provided dataset $D = \{(c_i, P(c_i))\}_{i=1}^{m}$, where each c_i represents a unique configuration in our configuration space C and $P(c_i)$ is the associated performance measurement. Applying Q to a subgroup $S \subseteq C$ yields a score $Q(S)$ that quantifies how exceptional the performance distribution within S is compared to the global distribution over C. Enumerating and scoring each subgroup then yields a set of real-valued *exceptionality measurements* $M = \{Q(c_i, P(c_i))\}_{i=1}^{m}$, which can be ranked to identify the most exceptional subgroups [3, 25].

There are multiple ways interesting measures can be defined and ranked. Common choices for numerical targets include mean-based measures, which compare subgroup means against the global mean, distribution-based measures, such as Kullback-Leibler (KL) divergence, assessing how much the target distribution of the subgroup diverges from the global distribution [7, 22], and rank-based measures, which evaluate whether the subgroup's target values tend to be larger or smaller than the complement according to nonparametric tests. Mann-Whitney U, for instance, assesses whether the ranks of subgroup observations are skewed, making it robust to outliers and distributional assumptions [3]. Each measure captures different aspects of how the target distribution within a subgroup deviates from the global distribution. As such, different analysis goals may benefit from a different choice of measure.

Since enumerating all possible subgroups is often infeasible, subgroup discovery algorithms typically employ search and pruning strategies to efficiently explore the space of possible subgroups and eliminate unpromising candidates early.

A commonly applied pruning scheme is *optimistic estimate pruning*, which, for a subset S, cheaply computes an optimistic estimate $Q'(S)$ that bounds the best possible quality any specialization of S could attain. If $Q'(S)$ is no greater than the current k-th best score, the

entire subtree below S can be safely skipped. For many mean- and rank-based measures this idea yields tight closed-form or one-pass bounds, which dramatically reduces the number of evaluated candidates while preserving exactness for top-k retrieval [25]. Pruning can also rely on other, complementary conditions. Minimal support thresholds remove trivially small subgroups that are unlikely to be meaningful and accelerate search. Statistical significance tests and multiple-testing corrections serve as post-hoc filters to reduce false positives. Redundancy control, for example by enforcing minimal rule length or by penalising overlap with already selected subgroups, yields more diverse result sets [3, 12].

Search strategies range from exhaustive to heuristic. Exhaustive methods systematically explore available options while maintaining a global top-k list of best subgroups. They are attractive because they can guarantee optimality with respect to a given quality function when paired with sound pruning [25]. Pattern-growth techniques, usually utilizing frequent-pattern trees, compress repeated selector co-occurrences to avoid repeated work when building refinements, which is useful when the number of selectors is large [23]. Heuristic searches, such as beam search or greedy refinement, restrict exploration to a small frontier of promising candidates and trade completeness for speed; they are commonly used when quick, interpretable rules are required. Evolutionary or stochastic search methods have also been applied to find diverse or multi-objective subgroup sets when the search space is too large for exhaustive approaches [3].

2.5 Similarity and Divergence Measures

To compare subgroups in terms of their structure and behaviour, we use two complementary classes of measures: (i) *set-based similarity metrics*, which give a concrete quantification of how much two subgroups overlap, and (ii) *distributional divergence metrics*, which compare how strongly the behaviour of a subgroup deviates from the global population. This section introduces the relevant metrics used throughout this thesis.

2.5.1 Precision, Recall, and F1 Score

Precision and *Recall* are fundamental similarity measures used across several domains, such as information retrieval, pattern mining, and machine learning [8]. For a given dataset D, we establish two subsets, a reference set $S_R \subseteq D$, representing the ground truth or target patterns, and an alternative set $S_A \subseteq D$, representing the patterns identified by an algorithm or method that is being evaluated. The overlap of both sets is then assessed in two complementary ways.

Recall expresses the proportion of elements of the reference set S_R that are successfully recovered by S_A:

$$\text{Recall}(S_A, S_R) = \frac{|S_A \cap S_R|}{|S_R|}.$$

Precision expresses the proportion of elements selected by S_A that truly belong to the reference set:

$$\text{Precision}(S_A, S_R) = \frac{|S_A \cap S_R|}{|S_A|}.$$

Each measure captures a different aspect of similarity: recall quantifies completeness with respect to a reference, while precision quantifies correctness of the selected set.

Because improvements in one do not necessarily imply improvements in the other, it is sometimes useful to combine them into a single score. The **F1 score** is the harmonic mean of precision and recall and provides a balanced measure, reflecting trends in both precision and recall:

$$\text{F1}(S_A, S_R) = 2\,\frac{\text{Precision}(S_A, S_R) \cdot \text{Recall}(S_A, S_R)}{\text{Precision}(S_A, S_R) + \text{Recall}(S_A, S_R)}.$$

These metrics are widely used for evaluating the similarity between discovered patterns, classifier outputs, or candidate subspaces, and they form an essential component for analysing structural agreement between different subsets of a given dataset.

2.5.2 Kullback–Leibler Divergence

Kullback–Leibler (KL) divergence is a measure from information theory that quantifies how one probability distribution differs from another [7, 22]. For two discrete distributions P and Q over a shared domain $\mathcal{X}$, it is defined as

$$\text{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}.$$

KL divergence quantifies the expected information loss incurred when a distribution Q is used to approximate a true distribution P. It can, for example, indicate how strongly a learned model distribution differs from the empirical data distribution, or how the behaviour of a subgroup of configurations deviates from that of the overall population. KL divergence is always non-negative, and equals zero if and only if $P = Q$. Therefore, a large KL divergence indicates that P assigns substantial probability to events that are rare or unlikely under Q, while small values indicate that P resembles Q closely.

In the context of this thesis, KL divergence is used to quantify how strongly the *performance distribution* of a subgroup deviates from that of the full configuration space. Let $P(c)$ denote the numeric performance variable measured for each configuration $c \in C$. Both the dataset and any subgroup induce empirical distributions over these performance values. We therefore define:

- P_{D}: the estimated performance distribution over all configurations in a dataset D, and
- P_{s}: the estimated performance distribution over the configurations contained in a given subgroup S.

The divergence $\text{KL}(P_{\text{s}} \parallel P_{\text{D}})$ thus measures how strongly the performance behaviour inside the subgroup differs from what would be expected under the dataset-wide performance distribution. In contrast to structural metrics, such as precision and recall, which

focus on *which configurations* appear in a subgroup, KL divergence characterises *how their performance values are distributed*. It thus captures shifts in central tendency, spread, and multi-modality, providing a complementary view on subgroup distinctiveness that extends beyond simple set overlap.

2.6 Syflow

SYFLOW is a novel approach to subgroup discovery that deviates from the classical enumerate-and-score paradigm. Instead, it utilises machine-learning techniques to formulate subgroup discovery as a continuous optimisation problem [32]. Central to this approach, SYFLOW defines subgroup membership through differentiable, soft selectors $\hat{\pi} : C \mapsto [0, 1]$, which assign each configuration $c \in C$ a membership degree in the interval $[0, 1]$. This allows the use of gradient-based optimisation methods to maximise a *weighted KL divergence* objective that captures distributional performance differences between the candidate subgroup and the overall population, ultimately discretising a subgroup membership that can be formalized into a human-interpretable rule.

Weighted KL Divergence For a subgroup of size $|S|$ and an exponent γ, the weighted divergence between the subgroup distribution P_{s} and the dataset-wide distribution P_{D} is defined as

$$\mathrm{WKL}_{\gamma}(P_{\mathrm{s}} \parallel P_{\mathrm{D}}) = |S|^{(\gamma-1)} * KL(P_{\mathrm{s}} \parallel P_{\mathrm{D}})$$

The multiplicative factor $|S|^{(\gamma-1)}$ serves as a support-weighting mechanism, discouraging very small subgroups, therefore preventing the optimisation from favouring sets of outliers, such as single configurations with noticeably divergent performance.

Regularizer A common issue in subgroup discovery is the selection of highly similar subgroups, as the *top-k* scoring subgroups often exhibit significant overlap. To mitigate this, SYFLOW introduces a regularizer, which influences the current objective based on the KL divergence between the current candidate subgroup and all previously selected subgroups. The parameter λ thereby controls the strength of this regularizer. Therefore, the final objective optimised by SYFLOW can be expressed as:

$$\mathrm{WKL}_{\gamma,\lambda}(P_{\mathrm{s}} \parallel P_{\mathrm{D}}) = |S|^{(\gamma-1)} * KL(P_{\mathrm{s}} \parallel P_{\mathrm{D}}) + \lambda \sum_{i=1}^{k-1} KL(P_{\mathrm{s}} \parallel P_{\mathrm{s}_i})$$

In this thesis, we use SYFLOW as the subgroup discovery algorithm of choice, to identify subgroups of datasets obtained from sampling highly configurable software systems and comparing them against the full configuration space.

3

Methodology

This chapter describes the experimental methodology used to evaluate how sampling affects subgroup discovery with SYFLOW. We adopt an empirical evaluation design in which subgroups discovered on sampled data are systematically compared to subgroups obtained from the full dataset. We begin by stating the research questions, followed by a description of how these questions are operationalised in terms of datasets, metrics, and experimental setup.

3.1 Research Questions

The main objective of this thesis is to investigate how different sampling strategies and sample sizes affect the quality and stability of subgroups discovered by SYFLOW. In particular, we analyse how well subgroups obtained from sampled datasets approximate those derived from the full configuration space, which serves as the reference point for our evaluation.

Our first research question examines how stable subgroup discovery results are with respect to the choice of sampling strategy. Since different sampling strategies may introduce unique biases or distortions, understanding their effect is essential for identifying strategies that yield subgroups most similar to those from the full dataset.

RQ1. *How do different sampling strategies affect the subgroups identified by Syflow compared to those found in the full dataset?*

Our second research question focuses on the role of sample size in determining the stability and similarity of subgroup discovery results. Smaller samples will generally lead to higher variance and SYFLOW may behave inconsistently when provided with limited information. Thus, we investigate how various sample sizes influence the subgroups found by SYFLOW across repeated random draws, and whether we can identify a common trend in how many samples are required to obtain stable and representative results.

RQ2. *How do different sample sizes affect the subgroups identified by Syflow compared to those found in the full dataset?*

Together, these research questions provide a comprehensive understanding of how the choice of sampling strategy and sample size impact subgroup discovery in configurable software systems. The insights gained from this analysis aim to guide practitioners in selecting sampling strategies and sample sizes that balance cost-efficiency with reliability.

System	Domain	|F|	|O|
7z	Compression	43	68,640
BerkeleyDBC	Database	18	2,560
Dune	PDE Solver	31	2,304
JavaGC	JVM GC Tuning	38	193,536
LLVM	Compiler Infrastructure	11	1,024
Polly	LLVM Polyhedral Optimizer	39	60,000
VP9	Video Codec	41	216,001
x264	Video Encoder	16	1,152

Table 3.1: Configuration space characteristics for each dataset D.

3.2 Operationalisation

To answer our research questions, we analyse a collection of more than 15 000 experimental results generated from combinations between eight datasets, five sampling strategies, five sample sizes, and 100 random seeds per sampled strategy. Each result contains information on precision, recall and weighted KL divergences between subgroups discovered on sample sets and their respective full dataset.

3.2.1 Datasets

We rely on eight real-world configurable software systems, originally introduced and collected by Kaltenecker et al. [19], and summarised in Table 3.1 together with their respective number of features and configurations. Each dataset represents its configuration space C as a binary vector,

$$C = [c_1, \ldots, c_n], \qquad c_j \in \{0, 1\}$$

where each binary variable indicates whether the corresponding feature is selected.

While this encoding is natural for Boolean features, several systems contain parameters that were *originally numeric*. These parameters are represented in the available configuration files using a standard *one-hot encoding*: each parameter with domain $\{v_0, v_1, \ldots, v_k\}$ is expanded into a set of mutually exclusive binary features,

$$x_{v_0}, x_{v_1}, \ldots, x_{v_k}$$

exactly one of which is set to 1 to indicate the chosen value. Thus, numeric features appear as small groups of one-hot encoded binary selectors instead of single-valued parameters.

Because Syflow expects each feature to be represented as a single numeric input, we preprocess these one-hot encoded feature groups. For every group of mutually exclusive candidate-value flags, we collapse the one-hot representation into a single numeric feature by assigning the corresponding value v_i of the active flag. If none of the flags are active, we assign a default value of 0.

Sampling Strategy	Sample Sizes / Parameters	Seeds
Random	$n \in \{1, 5, 10, 20, 50\}$	0–99
Distance	$n \in \{1, 5, 10, 20, 50\}$	0–99
Diversified-Distance	$n \in \{1, 5, 10, 20, 50\}$	0–99
SAT Solver-Based	$n \in \{1, 5, 10, 20, 50\}$	—
Combinatorial t-wise	$t \in \{2, 3, 4\}$	—

Table 3.2: Overview of sampling strategies, their parameters, where n denotes percent of the full dataset used, and t is the strength of combinatorial coverage, and seed configurations used in the experimental setup.

After preprocessing, every configuration $c \in C$ has a clean mixed numerical representation compatible with SYFLOW's input format. Each configuration is associated with a runtime performance measurement $P(c)$, which serves as the target variable for subgroup discovery.

3.2.2 Experimental Setup

Our experimental workflow consists of three main stages: sampling, subgroup discovery, and evaluation. These steps are explained in detail below.

Sampling. For each dataset, we construct sample sets according to the sampling strategy, sample size, and seed as listed in Table 3.2. Random, distance-based, and diversified distance-based sampling require randomness to generate variance across runs, and are therefore executed once for every combination of seed and sample size.

In contrast, our implementations of SAT solver-based sampling and combinatorial t-wise sampling are deterministic and do not depend on random seeds. Consequently, only a single sample is generated for each specified size or t-value.

Overall, this procedure yields 1,508 sample sets per dataset, or 12,064 sample sets across all datasets.

Subgroup discovery. For each sample set, as well as for each full dataset, we run SYFLOW with fixed hyperparameters $\lambda = 5$ and $\gamma = 0.3$. The full-dataset run produces a set of reference subgroups, in the following referred to as *full subgroups*, which serve as the baseline for comparison. Each sampled run generates a corresponding set of *sampled subgroups* which may differ in rule structure, configuration coverage, or outcome behaviour.

To avoid degenerate results, we ensure that all discovered subgroups within a run are unique. We enforce uniqueness by removing any subgroup whose configuration set is identical to that of a previously discovered subgroup. This results in up to five distinct subgroups per run.

Evaluation. The final stage consists of comparing the subgroups discovered on sampled data with those obtained from the full dataset. We evaluate two forms of similarity:

- **Structural similarity**: using Recall and Precision
- **Performance-distribution divergence**: using weighted KL divergence

Before computing structural similarity, an important practical issue must be addressed. A direct comparison between full-data and sampled subgroups is problematic because subgroups discovered on the full dataset may contain many configurations that are not part of the sampled dataset. Comparing them directly would artificially diminish structural similarity, since the mismatch would reflect differences in the available configuration space rather than genuine differences in subgroup structure.

To address this, we introduce a reduction step: for every full-dataset subgroup S_{full}, we restrict it to those configurations that also appear in the sample set D_{sample}:

$$S_{\text{full}}^{\downarrow} = S_{\text{full}} \cap D_{\text{sample}}.$$

All structural metrics are computed between the sampled subgroup S_{sample} and the reduced full subgroup $S_{\text{full}}^{\downarrow}$. This ensures that both subgroups are evaluated on a common configuration space and that differences reflect genuine structural divergences rather than sampling artefacts.

In addition to structural similarity, we also evaluate performance-based similarity between subgroups by analysing how strongly the performance distribution of a subgroup deviates from that of the population using a discrete, weighted KL divergence. These measurements serve to evaluate whether SYFLOW's optimisation objective behaves consistently across the full dataset and its sampled subsets. This is particularly relevant when a subgroup discovered on the full dataset is only partially present, or missing entirely, in the sampled dataset; such cases may indicate that sampling has removed or distorted performance-relevant regions of the configuration space, therefore rendering SYFLOW unable to recover the same subgroup in both datasets.

Since we are interested in answering why certain subgroups are not recovered in sampled datasets, we compute two complementary KL divergences for each subgroup pair:

- the sampled subgroup evaluated within the sampled dataset, and
- the (reduced) full subgroup evaluated within the sampled dataset.

For each case, we compute the weighted divergence $\mathrm{KL}_{\gamma}(D, S)$ between the empirical performance distribution of the dataset D and that of the subgroup S. The exponent γ matches SYFLOW's configuration to maintain a consistent treatment of subgroup size.

At this point, we have obtained metric values for every pairwise combination of sampled and full subgroups. However, because subgroups are guaranteed to be unique within each run, each sampled subgroup can meaningfully correspond to at most one full-data subgroup. A direct row-wise comparison is similarly infeasible, as subgroup discovery is not consistent in either the ordering or the number of subgroups found. Thus, comparing the k-th sampled subgroup with the k-th full subgroup would be arbitrary and uninformative.

To obtain a principled and interpretable basis for evaluation, we therefore establish a one-to-one correspondence between sampled and full-data subgroups. For each sampled

subgroup S_{sample}, we identify the full-data subgroup whose reduced form $T^{\downarrow}$ maximises configuration space overlap according to the F1 score:

$$S_{\text{match}} = \arg\max_{T \in \mathcal{S}_{\text{full}}} \text{F1}\big(S_{\text{sample}}, T^{\downarrow}\big).$$

The F1 score serves as the matching criterion, as it jointly accounts for both precision and recall in a single measure, avoiding bias towards either. Ties are resolved using a first-come, first-served strategy, ensuring that each full subgroup can be matched at most once. If the two runs yield different numbers of subgroups, any remaining unmatched subgroups are excluded from further analysis.

This matching step reduces the otherwise arbitrary set of up to 25 pairwise comparisons to a single, semantically justified alignment per subgroup found in the sample set. It ensures that all computed metrics, structural similarity as well as weighted KL divergences, compare only corresponding subgroups rather than unrelated ones.

3.3 Data Analysis

The procedures described in the previous sections yield, for every sampled run, a matched set of similarity measures and weighted KL divergence values comparing sampled subgroups to their corresponding full-dataset subgroups. This section outlines how these measurements are aggregated and analysed to answer the research questions.

KL Divergence We introduce one final computation step specific to the weighted KL divergence metrics. In section 3.2.2, we described how we compute weighted KL divergence values for each matched subgroup pair. However, since KL divergence on its own is difficult to interpret, we instead focus on the *difference* in KL divergence between sampled and full subgroups within the same dataset. Therefore, for each data pair $(S_{\text{sample}}, S_{\text{full}})$, we compute:

$$\Delta\text{KL}_{\gamma}(D, S) = |\text{KL}_{\gamma}(D, S_{\text{sample}}) - \text{KL}_{\gamma}(D, S_{\text{full}})|.$$

This way, we can evaluate how much more (or less) the sampled subgroup diverges from the population compared to the full subgroup when evaluated within the same dataset D. A value close to zero indicates that the sampled subgroup captures performance distribution similarly, while larger values express that the performance distribution within the sample deviates more strongly from the population than the full subgroup, making recovery of the same subgroup within both datasets less likely.

3.3.1 RQ1: Effect of Sampling Strategy

The first research question investigates how different sampling strategies affect the quality of subgroups discovered by SYFLOW. We fix the sample size to 20% of the full dataset and analyse results grouped by sampling strategy. For t-wise sampling, each value of t yields a different fixed sample size depending on the dataset; we therefore treat each t as a separate strategy and observe them independently. For each dataset, we then analyse the distribution

of structural similarity scores (Precision, Recall) and plot ΔKL_γ against its respective $F1$ score.

Together, we aim to identify how sensitive each strategy is to stochastic effects, and which strategies most consistently yield subgroups that most closely resemble those from the full dataset. Additionally, KL divergence values help us understand whether certain strategies systematically distort performance distributions, thereby affecting SYFLOW's ability to recover similar subgroups or if failure to recover similar subgroups is due to internal algorithmic distortions within SYFLOW itself.

3.3.2 RQ2: Effect of Sample Size and Stability

To answer **RQ2**, we analyse how subgroup quality and stability evolve as the sample size increases. In this analysis, we focus on random sampling as a representative baseline strategy. Random sampling allows sample size to be varied independently, making it well-suited for studying subgroup discovery under various sample sizes. While results for distance-based and diversified distance-based sampling have been omitted from our evaluation due to technical issues in our implementation, their partial results are reported in the Appendix A.

We investigate whether structural similarity measures (precision, recall, and F1 score) improve consistently as the sample size increases from 1% to 50% of the full dataset, or whether improvements plateau beyond certain thresholds. We further examine whether these thresholds are consistent across datasets with different configuration space sizes and to what extent increasing sample size reduces variability across repeated runs.

In addition, we analyse how ΔKL_γ evolves as sample size increases, to assess whether sampling-induced distortions in performance distributions diminish with larger samples, or whether such effects persist independently of the amount of available data.

Overall, this analysis provides insight into how much data is required for reliable subgroup discovery on performance data on configurable software systems and how the effectiveness of each sampling strategy depends on the amount of available measurements.

4

Evaluation

This chapter presents the experimental results of our study. We analyse how different sampling strategies and sample sizes influence the subgroups discovered by SYFLOW, focusing on both structural similarity to reference subgroups and differences in performance-distribution behaviour.

Our evaluation is organised around the two research questions presented in Chapter 3. For each research question, we report aggregated results across datasets, sampling runs, and random seeds in order to highlight systematic trends as well as variability across configurations. A detailed discussion of the findings follows in Section 4.2, together with an assessment of threats to validity in Section 4.3.

4.1 Results

Following the methodology outlined in Chapter 3, we present the results of our experiments structured around the two research questions. Beginning with an analysis of how different sampling strategies impact subgroup discovery quality, we then explore the effects of varying sample sizes on the stability and fidelity of discovered subgroups.

4.1.1 RQ1: Sampling Strategy

We address our first research question by analysing how different sampling strategies influence the quality of discovered subgroups. Here, structural similarity is assessed using precision and recall, capturing how well sampled subgroups recover the configuration sets of their full-data counterparts. In addition, we examine weighted KL divergence to assess whether the performance distributions between the datasets and the respective subgroups identified by SYFLOW remain consistent under different sampling strategies. By comparing the distributions of these measures across datasets and seeds, we evaluate which strategies yield stable and reliable subgroup discovery outcomes.

Reference Subgroup Characteristics Before analysing structural similarity, we first characterise the reference subgroups, i.e., the subgroups found in the full datasets, themselves. Table 4.1 summarises key structural and distributional properties of the subgroups discovered by SYFLOW on the full datasets. For each system, values are reported as ranges across the (up to) five subgroups identified, capturing variability in rule complexity, subgroup coverage, and performance distinctiveness. Overall, the datasets exhibit comparable

Dataset	|O|	|S|	Clauses	Subgroup Size	Relative Size	KL Divergence
7z	68640	4	1 – 4	6864 – 27456	10.0% – 40.0%	2.73 – 7.37
BerkeleyDBC	2560	4	2 – 2	512 – 1024	20.0% – 40.0%	1.29 – 11.54
Dune	2304	5	3 – 4	280 – 560	12.2% – 24.3%	2.19 – 8.49
JavaGC	193536	4	2 – 3	16128 – 80640	8.3% – 41.7%	0.93 – 6.28
LLVM	1024	4	3 – 4	64 – 128	6.2% – 12.5%	2.34 – 11.82
Polly	60000	5	2 – 4	5000 – 20000	8.3% – 33.3%	0.86 – 19.56
VP9	216000	5	1 – 4	21600 – 108000	10.0% – 50.0%	1.07 – 10.15
x264	1152	5	1 – 3	96 – 768	8.3% – 66.7%	0.54 – 6.16

Table 4.1: Metadata of subgroups discovered across full datasets. Values are reported as ranges across the subgroups found per dataset.

structural characteristics, with discovered subgroups covering between roughly 5% and 67% of the respective configuration spaces and rule complexity ranging from very compact descriptions consisting of a single clause to more elaborate rules with up to four clauses. Weighted KL divergence values likewise span a wide range, indicating that some datasets contain subgroups with strongly distinct performance distributions, while others exhibit more subtle distributional differences without pronounced outliers.

Structural Similarity We begin our assessment of sampling strategies by examining structural similarity between subgroups discovered on sampled data and those from the full configuration space. To this end, we analyse precision and recall metrics across all datasets and sampling strategies, reporting median values and interquartile ranges where applicable. In the case of LLVM, we discard t-wise sampling results, as its configuration space does not carry any restrictions, and therefore the resulting sampled set size for any t-wise strategy is exactly one, making meaningful subgroup discovery impossible. Another important detail is that combinatorial t-wise sampling and solver-based sampling are deterministic and therefore only executed once per sample size and dataset. As a result, at most five reference subgroups are obtained per dataset and sampling strategy. Accordingly, for deterministic strategies we analyse each matched subgroup individually, rather than reporting aggregated statistics as is done for stochastic sampling strategies. Additionally, for x264 and BerkeleyDBC, the sample sets generated for $t = 3, 4$ are identical. This is due to the small number of configuration options in these systems (Table 3.1), which limits the number of distinct interactions that can be covered. As a consequence, SyFlow discovers identical subgroups for these values of t.

Figure 4.1 presents the distribution of precision values across sampling strategies for each dataset. Across most systems, stochastic sampling strategies (random, distance-based, and diversified distance-based) achieve higher median precision values than combinatorial and solver-based approaches.

In particular, random sampling exhibits median precision values above 0.8 for the majority of datasets and reaches precision scores of 1.0 in four out of the eight systems. In contrast, the precision distributions also show wide interquartile ranges. This high variability across runs indicates a strong sensitivity to the specific sampled configurations, suggesting that

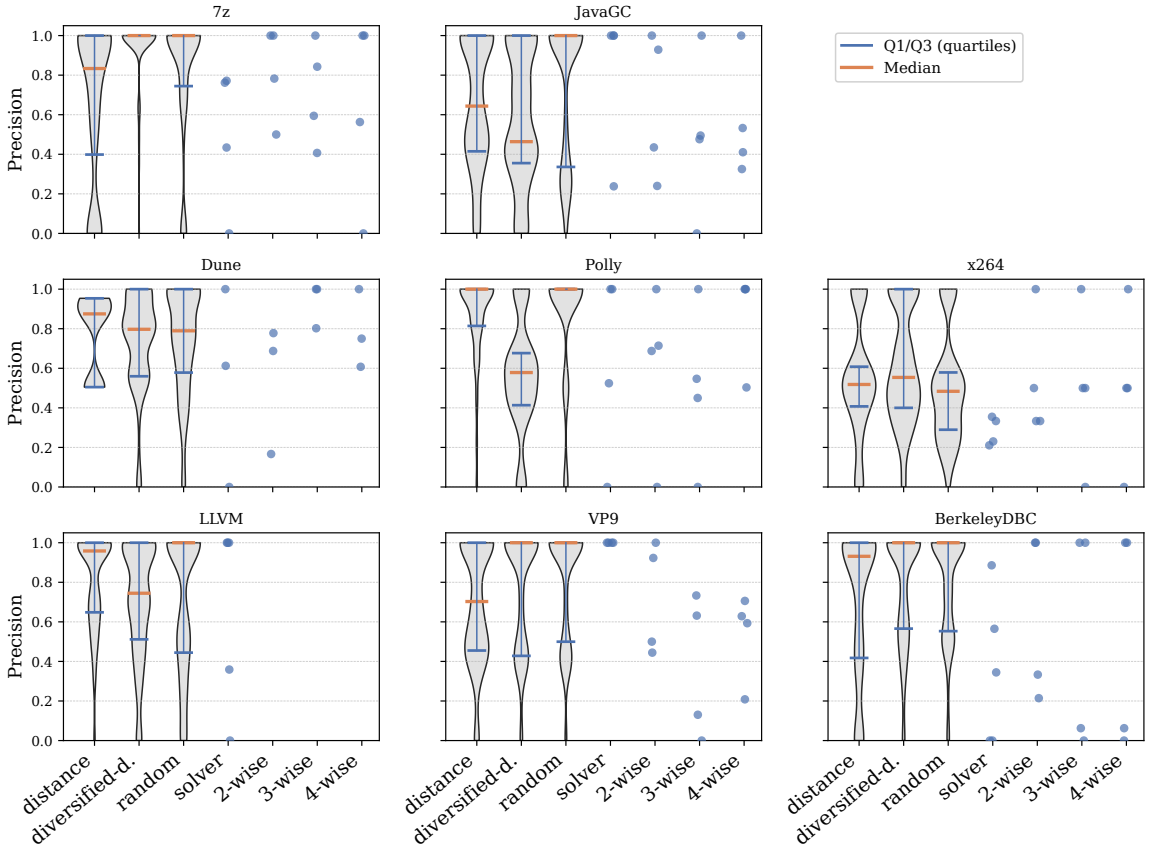

Figure 4.1: Precision distributions across sampling strategies for all datasets.

high precision is not consistently achieved. Meanwhile, distance-based sampling reduces this variability, as reflected by narrower interquartile ranges compared to random sampling. In several datasets, this stabilisation is accompanied by slightly higher median precision values, while in others the median precision decreases. This shows that reduced variability across runs does not necessarily lead to better structural similarity. Its diversified variant further amplifies this effect with mixed results, reducing variability in some cases and lowering median precision values in others.

Deterministic strategies perform poorly in terms of recovering structurally similar subgroups. Both t-wise and solver-based sampling often manage to recover a reference subgroup with perfect precision, however, these cases are isolated and typically limited to a single subgroup per dataset, with the remaining discovered subgroups exhibiting substantially lower precision values, frequently approaching zero. This behaviour is consistent regardless of sample sizes, characteristics of the reference subgroups, or, in case of t-wise sampling, the value of t.

Recall values, shown in Figure 4.2, are generally higher than precision values across sampling strategies, with several datasets exhibiting median recall values close to 1.0.

Looking at stochastic strategies first, distance-based approaches continue to perform well, achieving perfect median recall values of 1.0 in five out of eight datasets, with diversified distance-based sampling closely following this trend and interquartile ranges that are

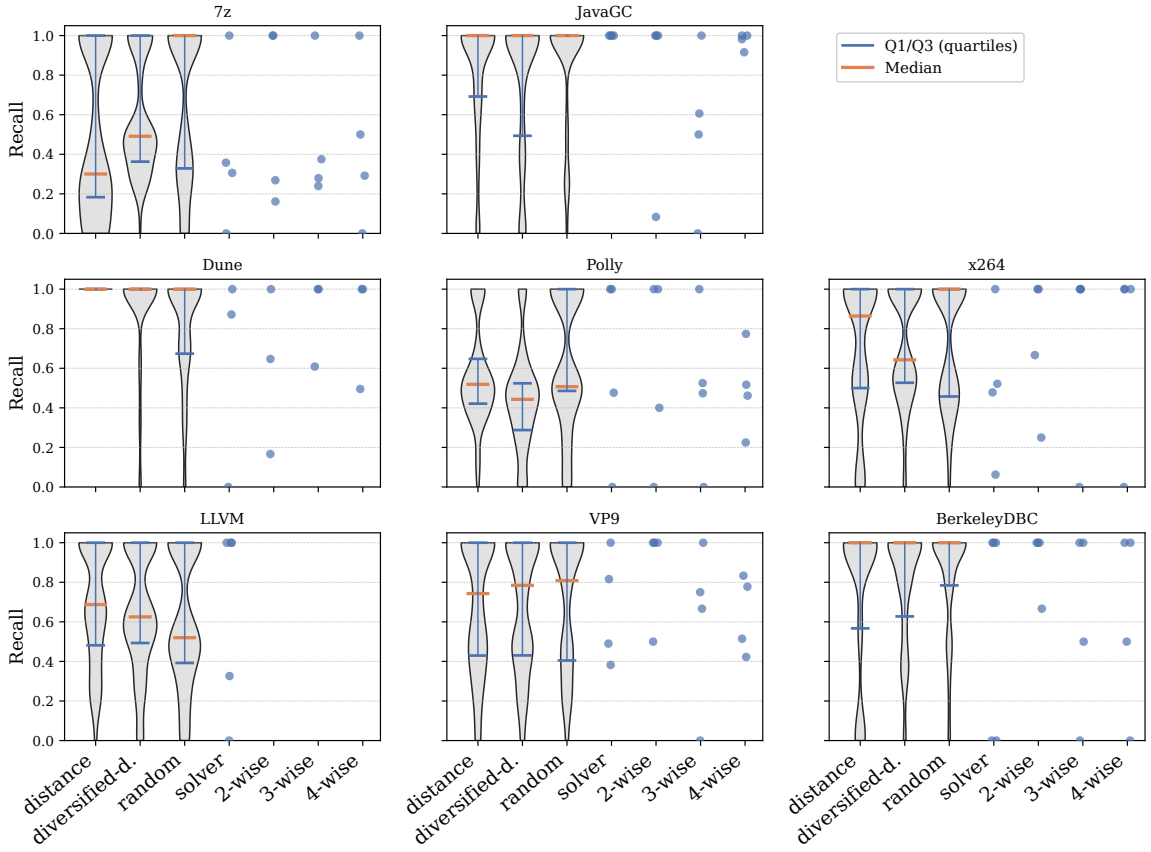

Figure 4.2: Recall distributions across sampling strategies for all datasets.

typically narrower than random sampling. In the Dune dataset, distance-based sampling even manages to achieve a perfect recall score of 1.0 across all discovered subgroups. Random sampling on the other hand outperforms both distance-based strategies in terms of median recall values across all but one dataset, and achieving a perfect median recall value of 1.0 in seven out of the eight systems tested.

Meanwhile, deterministic strategies exhibit highly variable recall performance. Similar to precision, both t-wise and solver-based sampling can enable Syflow to recover at least one reference subgroup with high recall. However, the remaining discovered subgroups typically show a wide range of recall values, spanning from near-perfect recall to values close to zero.

Even in systems such as JavaGC, where high recall values are commonly achieved across all sampling strategies, t-wise sampling still produces highly heterogeneous sets of discovered subgroups. In particular, while some subgroups attain near-perfect recall, others exhibit recall values close to zero, indicating that not all reference subgroups are recovered equally well. Thus, deterministic strategies may recover individual reference subgroups accurately, but they do not reliably preserve the broader subgroup structure present in the full dataset.

Jointly observing precision (Fig. 4.1) and recall (Fig. 4.2) shows that recall values are frequently higher than precision values across datasets and sampling strategies. This

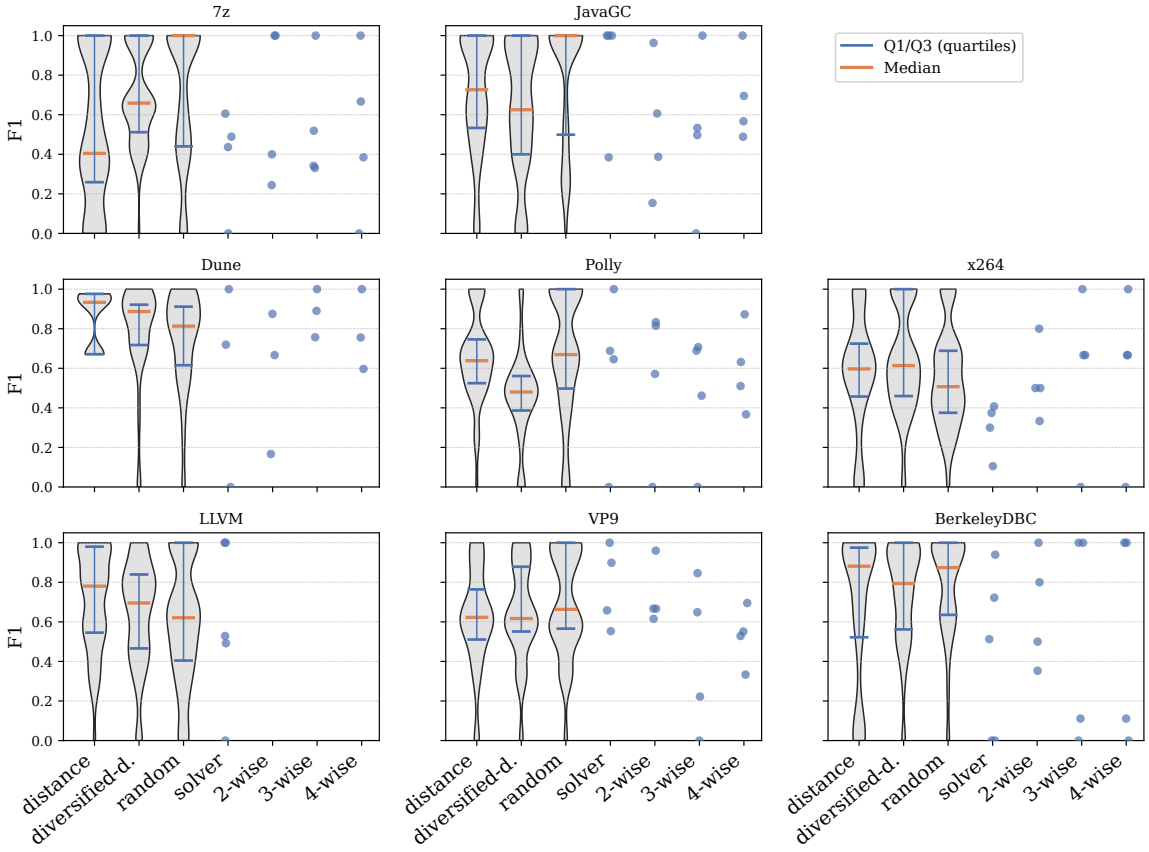

Figure 4.3: F1-score distributions across sampling strategies for all datasets.

indicates a tendency for sampling strategies to recover supersets of reference subgroups, trading selectivity for coverage. This trade-off is particularly pronounced in datasets such as x264 and JavaGC, where recall values consistently exceed precision values across most sampling strategies, with the exception of 3-wise sampling.

Figure 4.3 summarises structural similarity using the F1 score. Compared to precision and recall individually, F1 scores exhibit reduced separation between sampling strategies across most datasets, reflecting the previously observed trade-off between precision losses and recall gains.

Random sampling achieves high median F1 scores across datasets. However, this is often accompanied by substantial variability, with interquartile ranges frequently wider than those observed for distance-based strategies. Distance-based sampling improves on random sampling by reducing this variability while maintaining comparable F1 scores, with the exceptions of 7z and JavaGC. In these datasets, random sampling achieves a median F1 score of 1.0, whereas distance-based sampling attains median values of 0.4 and 0.6, respectively. Diversified distance-based sampling performs worse than its non-diversified variant, either through increased variability, reduced median F1 scores, or both.

Solver-based sampling occasionally enables SyFLOW to recover an individual reference subgroup with very high or even perfect F1 score. This behaviour can be observed in several datasets, including JavaGC, VP9, and Polly, where a small number of matched

subgroups cluster near the maximum F1 value. Nevertheless, this clustering is typically limited to one or very few subgroups, while the remaining discovered subgroups exhibit substantially lower F1 scores. As a result, solver-based sampling does not preserve the overall subgroup structure of the full dataset, and instead is more suited towards recovering isolated qualitative matches.

In contrast, when examining the relationship between structural similarity and the exceptionality of the corresponding reference subgroups, solver-based sampling exhibits inconsistent behaviour. Based on a manual inspection of the matched subgroups across datasets, subgroups discovered with the highest F1 scores are not necessarily those associated with the most distinctive performance patterns, and in some cases even correspond to reference subgroups with comparatively low exceptionality. This indicates that while solver-based sampling may recover structurally similar subgroups, these matches do not consistently reflect the most distributionally exceptional reference subgroups.

Combinatorial t-wise sampling shows similarly mixed behaviour. Across all values of t, individual subgroups may achieve high F1 scores. However, these cases are isolated and strongly dataset-dependent, while many other matched subgroups show little to no structural similarity. Increasing the interaction strength t does not lead to a consistent improvement in subgroup recovery, nor does it increase the number of reference subgroups recovered by SYFLOW. Consequently, while t-wise sampling can recover individual reference subgroups, it fails to reliably recover multiple subgroups simultaneously and does not robustly preserve the structural characteristics of the full subgroup set.

Moreover, subgroups recovered with high structural similarity are not consistently associated with reference subgroups exhibiting particularly distinctive performance behaviour. Across datasets, structurally well-matched subgroups may correspond to reference subgroups with comparatively low exceptionality, and increasing the interaction strength t does not lead to a more consistent alignment between subgroup structure and performance behaviour. This further supports the observation that, similar to solver-based sampling, t-wise sampling is able to recover individual reference subgroups, but fails to reliably preserve the relative importance or exceptionality of subgroups present in the full configuration space.

Distributional Similarity Figure 4.4 relates structural similarity, measured by the F1 score, to differences in weighted KL divergence between sampled and full-data subgroups evaluated within their respective sampled datasets. Solver-based and combinatorial t-wise sampling strategies are excluded from these plots due to their very limited number of discovered subgroups, which renders distributional trends difficult to interpret reliably.

A first observation is that KL divergence differences are all following a downward trend as F1 scores increase. This is to be expected, as subgroups that are structurally identical are inherently also identical in terms of their performance distributions, resulting in equal KL divergence values. As F1 scores decrease, the difference between KL divergences increases. This is, in so far, an intuitive consequence of the previous observation, however the relationship is not strictly linear. Instead, the majority of points form a dense, box-like region spanning approximately F1 values between 0.0 and 0.8 and KL differences between 0 and 10. While extreme outliers become more frequent at lower F1 scores, the central mass of points remains concentrated within this region.

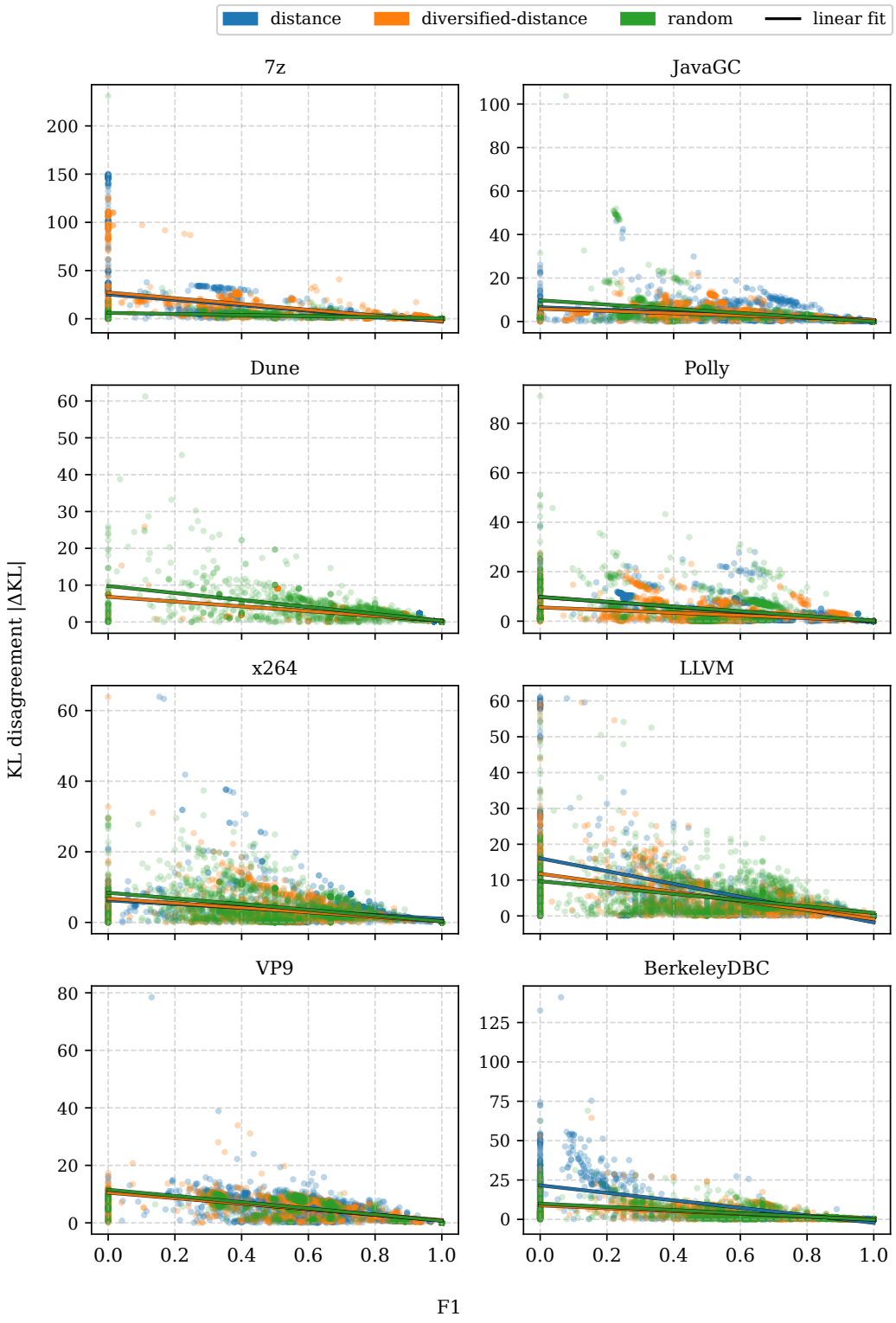

Figure 4.4: Weighted KL divergence difference distributions across sampling strategies for all datasets.

When comparing sampling strategies, no substantial differences are observed in the overall structure of the KL–F1 relationship. All observed strategies populate largely the same regions of the KL–F1 space, indicating similar relationships between structural similarity and performance-distribution divergence.

Random sampling exhibits higher dispersion in KL divergence differences, as reflected by a wider distribution of points across the KL–F1 space. In contrast, distance-based and diversified distance-based sampling yield more concentrated point distributions, with only minor differences in dispersion that vary across datasets and do not follow a consistent pattern.

4.1.2 RQ2: Sample Size

Dataset	1%	5%	10%	20%	50%
7z	686	3432	6864	13728	34320
BerkeleyDBC	25	128	256	512	1280
Dune	23	115	230	460	1152
JavaGC	1935	9676	19353	38707	96768
LLVM	10	51	102	204	512
Polly	600	3000	6000	12000	30000
VP9	2160	10800	21600	43200	108000
x264	11	57	115	230	576

Table 4.2: Absolute sample sizes corresponding to relative sampling budgets for each dataset.

To address our second research question, which examines the effect of sample size on subgroup quality and stability, we analyse how structural similarity and performance-distribution divergence evolve as sample size relative to the full dataset increases. This analysis allows us to identify whether improvements in subgroup recovery are gradual or if they stagnate at certain sample sizes, and to what extent increased sample sizes reduce variability across repeated runs.

Table 4.2 summarises the absolute sample sizes corresponding to the relative sampling budgets used in our experiments for each dataset. As configuration space sizes vary substantially between systems, identical percentage-based budgets can result in vastly different numbers of sampled configurations. This difference is particularly pronounced for small configuration spaces such as LLVM and x264, where even the smallest budgets correspond to only a handful of configurations.

Structural Similarity Figure 4.5 presents the distribution of *precision* values across different sample sizes for the random sampling strategy. Several datasets, such as Dune and JavaGC, exhibit an overall upward trend in median precision as sample size increases. In contrast, datasets such as BerkeleyDBC already achieve high median precision at very small sample sizes, with only marginal changes thereafter.

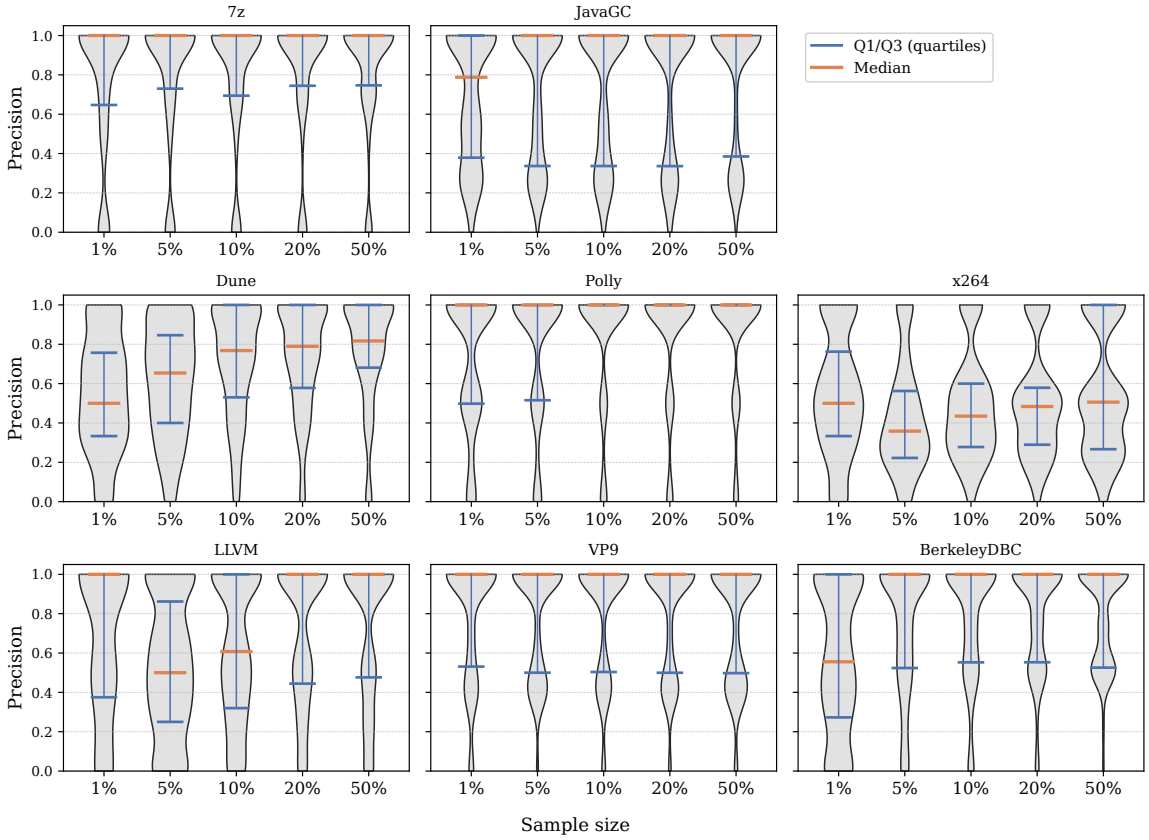

Figure 4.5: Precision distributions across sample sizes for random sampling strategy for all datasets.

For x264 and LLVM, precision decreases when increasing the sample size from 1% to 5% and 10%, before stabilising again at larger sample sizes. Although this behaviour appears counterintuitive, it is likely attributable to the extremely small absolute sample sizes at 1% (Table 4.2). With so few configurations selected, a small number of randomly chosen samples may, by chance, already cover most or all configurations of a reference subgroup, resulting in artificially high precision values. As the sample size increases slightly, additional configurations introduce variability without yet providing sufficient coverage to reliably recover subgroup structure, which can lead to temporarily lower precision.

Overall, increasing the sample size tends to reduce variability in precision across runs, although improvements in median precision are neither monotonic nor consistent across all datasets. Rather than exhibiting a clear plateau, datasets differ in the sample sizes at which precision stabilises, reflecting differences in configuration space size and subgroup complexity. Accordingly, for small configuration spaces such as LLVM and x264, precision stabilisation is only observed at larger sampling budgets, whereas larger systems such as VP9 and JavaGC already exhibit high median precision at relatively small sample sizes.

A similar pattern can be observed for *recall*, as shown in Figure 4.6. As observed for precision, datasets exhibit varying sensitivity to sampling size, with particularly high variability observed for sample sizes below 20%.

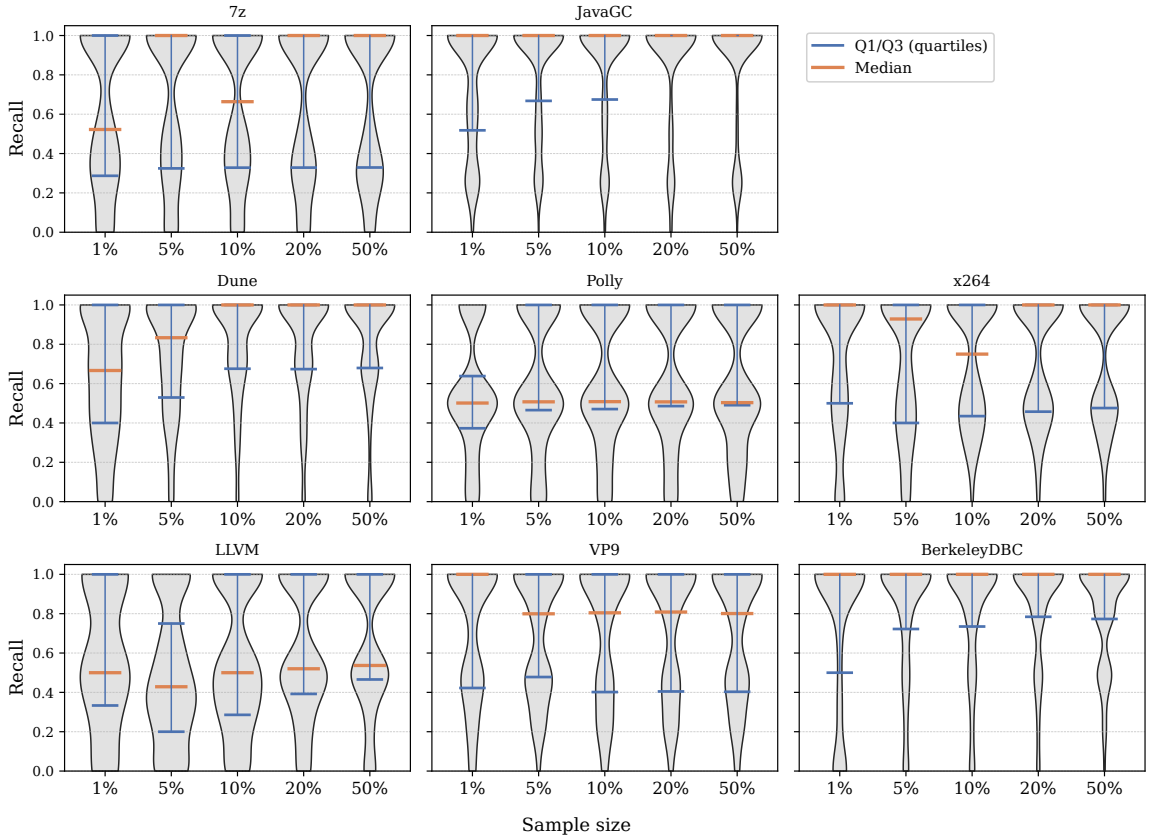

Figure 4.6: Recall distributions across sample sizes for random sampling strategy for all datasets.

For instance, x264 continues to show a decreasing trend in recall from 1% to 10% before stabilising at larger sample sizes, while 7z achieves its highest median recall at 5%, with neighbouring sample sizes yielding lower values before stabilising again at 20%. For larger datasets, VP9 shows a slight decrease from 1% to 5%, stabilising thereafter, whereas JavaGC primarily benefits from reduced variability, with interquartile ranges narrowing up to a sample size of 20%. Dune exhibits a monotonic increase in recall as sample size grows, while the remaining datasets maintain largely stable recall distributions across all sample sizes, with only minor changes in variability.

Taken together, these observations largely support the trends identified for precision, with very small sample sizes yielding notable instability in recall values across datasets, resulting in either increases or decreases in median recall depending on the specific dataset. This variability diminishes as sample sizes grow, with larger datasets typically achieving a stable recall value sooner than smaller ones.

Figure 4.7 summarises structural similarity using the F1 score across sample sizes for random sampling. The trends observed for precision and recall largely carry over to the F1 score, with datasets exhibiting varying sensitivity to small sampling sizes. Small datasets show fluctuating F1 scores at small sample sizes before stabilising at around 20%. Meanwhile, larger datasets stabilize already at 5%, while medium sized ones vary between

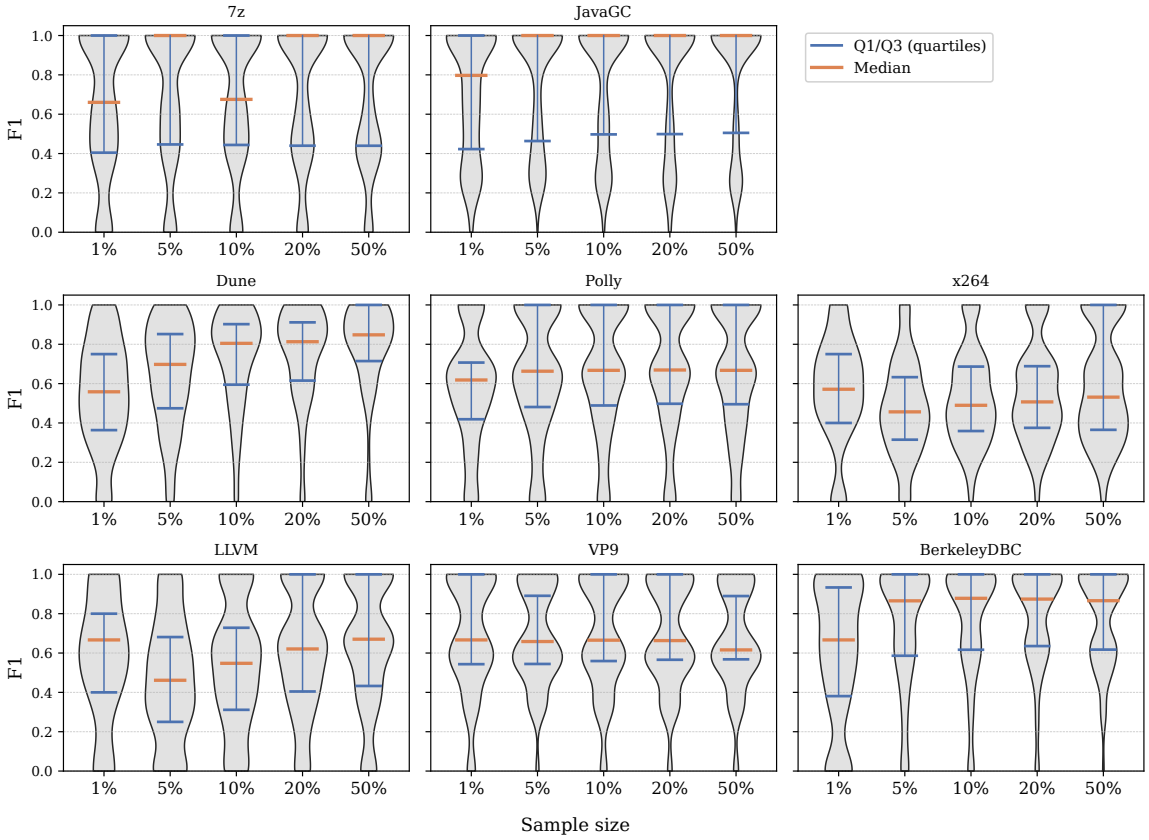

Figure 4.7: F1-score distributions across sample sizes for random sampling strategy for all datasets.

these extremes. Beyond these thresholds, there are diminishing returns from increasing the sampling size for random sampling, independent of absolute sample size increases.

Distributional Similarity Figure 4.8 relates structural similarity, measured by the F1 score, to differences in weighted KL divergence across increasing sample sizes for random sampling. Overall, the same characteristic relationship observed in Section 4.1.1 persists.

Notably, increasing the sample size does not fundamentally alter this behaviour. Subgroups discovered from both small and large samples populate the same regions of the KL–F1 space, and no qualitatively new patterns emerge as sample size increases. Instead, the primary effect of larger sample sizes is a reduction in variance, with points concentrating more tightly and at higher density towards lower KL divergence differences, while smaller sample sizes tend to exhibit a wider spread. Nevertheless, the general trend remains consistent across all sample sizes, with the exception of LLVM and 7z, where smaller samples show a slightly higher KL divergence spread at lower F1 scores.

In summary, increasing the sampling size primarily reduces the frequency of extreme outliers in the KL–F1 space, rather than altering the overall shape or location of the distribution. While larger samples lead to more tightly clustered results, they do not fundamentally change the relationship between structural similarity and performance-distribution divergence. This indicates that increasing sample size improves the stability of

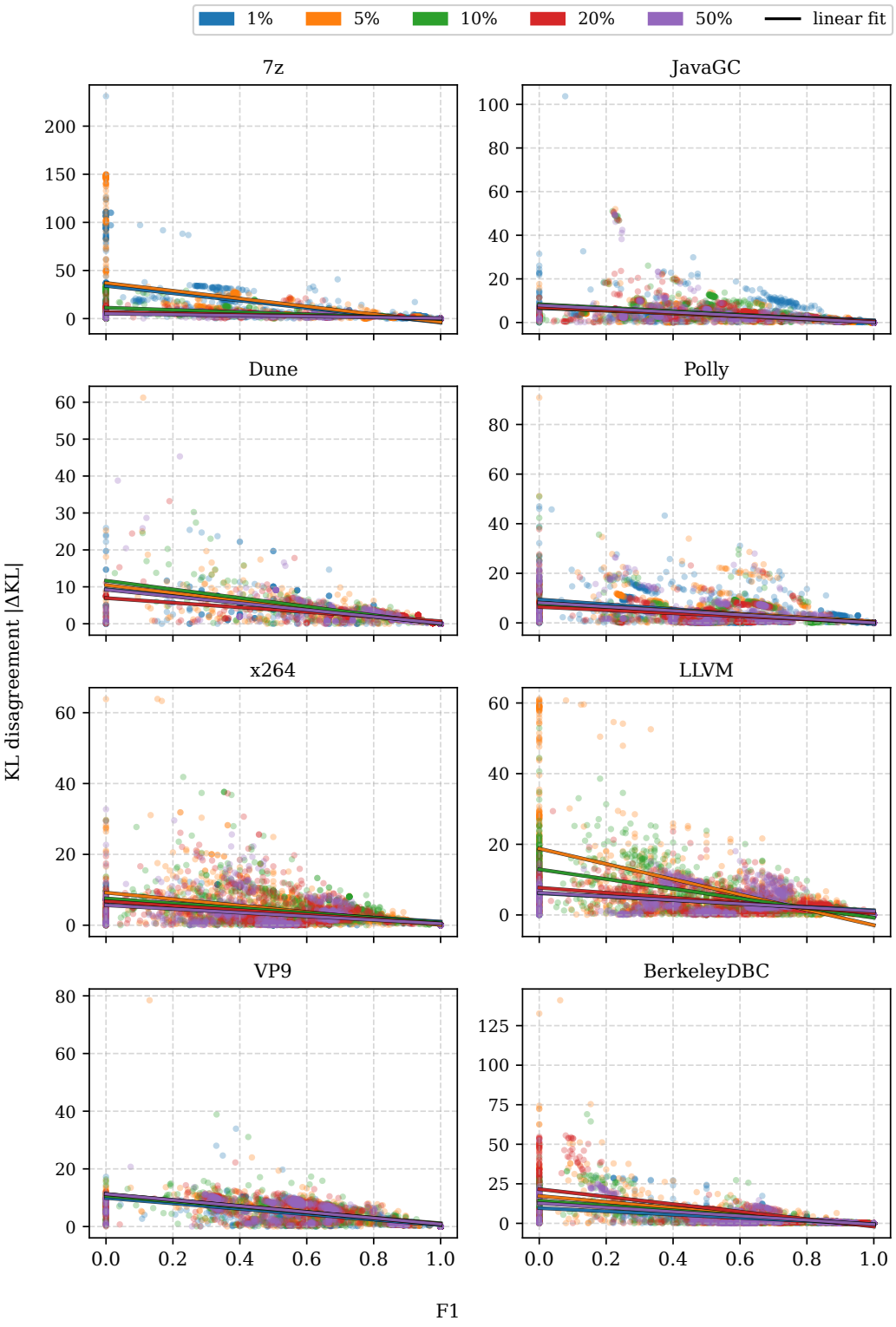

Figure 4.8: Weighted KL divergence difference distributions across sample sizes for random sampling strategy for all datasets.

distributional estimates, but does not reliably eliminate sampling-induced distortions in subgroup performance distributions.

4.2 Discussion

This section discusses and contextualises the experimental results by analysing how different sampling strategies interact with subgroup discovery in *Syflow*, and what this implies for the quality and reliability of the discovered subgroups.

4.2.1 RQ1: Sampling Strategy

When comparing different sampling strategies with respect to their ability to support subgroup discovery on sampled data, stochastic strategies (random, distance-based, and diversified distance-based) tend to produce sample sets from which SYFLOW discovers multiple subgroups with moderate to high structural similarity to the reference (for more details, see Appendix A.1). Meanwhile, deterministic strategies often recover only a small number of subgroups, sometimes with very high similarity.

As shown in Figure 4.3, stochastic strategies more frequently yield higher median F1 scores accompanied by narrower interquartile ranges, reflecting greater consistency across runs. In contrast, deterministic strategies exhibit pronounced dataset-dependent behaviour and wide spreads in subgroup similarity, largely driven by the fact that individual reference subgroups may be recovered well while others are missed entirely. As a result, aggregate similarity metrics are less comparable across strategies, and conclusions regarding relative performance must be drawn cautiously.

Among the stochastic approaches, random sampling frequently produces sampled configuration sets that allow SYFLOW to discover subgroups with high structural similarity to the reference subgroups. That said, this performance comes at the cost of substantial variability, reflected in wide spreads across runs and datasets. In practice, mitigating this variability would require either repeated sampling and reruns or substantially larger sample sizes, both of which increase measurement effort and limit the practical efficiency of pure random sampling.

Distance-based sampling and its diversified variant provide a more favourable trade-off. Both preserve high structural similarity while substantially reducing variability in the discovered subgroups. Distance-based sampling in particular often achieves high similarity comparable to random sampling, but with markedly more stable results, whereas the diversified variant further reduces variance at the cost of slightly less reliable subgroup recovery.

Deterministic strategies, such as combinatorial t-wise sampling and our implementation of solver-based sampling, reliably allow SYFLOW to recover at least one reference subgroup with high structural similarity in most datasets. In case of solver-based sampling in particular, SYFLOW typically manages to find multiple subgroups closely resembling the target subgroup. However, beyond these top-performing subgroups, the remaining discovered subgroups often exhibit substantially lower structural similarity. Therefore, the overall

reliability of solver-based sampling on its own remains limited without additional tests to identify which subgroups are likely to be recovered. Instead, solver-based sampling may be better suited as a complementary strategy to refine or validate subgroups discovered through stochastic sampling.

Combinatorial t-wise sampling shows highly inconsistent behaviour across datasets and interaction strengths t. Part of this may be attributed to the specific implementation of t-wise sampling used in this study, as it does not aim to cover all possible value combinations of interacting features. Instead, it focuses on interactions where all involved features are enabled simultaneously. While this design choice is common in configurable systems testing, it can be particularly problematic for subgroup discovery, where subgroup definitions may rely equally on disabled features or mixed-value interactions. As a result, structurally relevant subgroups whose defining characteristics involve such combinations may be systematically underrepresented or entirely absent from the sampled configuration sets. This bias provides an explanation for the inconsistent structural similarity observed for t-wise sampling and highlights a fundamental limitation of applying standard t-wise sampling techniques directly to subgroup discovery tasks and making it unsuitable as a standalone sampling strategy in this context, while underperforming as a potential validation-tactic in place of solver-based sampling.

Taken together, these observations suggest that the occasional recovery of highly similar subgroups by deterministic strategies should be interpreted with caution. Across datasets, structurally well-matched subgroups are not consistently among those exhibiting the most distinctive performance behaviour. Thus, while deterministic strategies can recover isolated matches, they do not reliably preserve the relative importance or ordering of subgroups present in the full configuration space.

Figure 4.4 shows that low KL divergence values do not reliably correspond to high F1 scores, meaning that subgroups discovered from sampled data can exhibit similar performance distributions despite low structural similarity to the reference, while structurally similar subgroups may still display noticeable distributional differences. This relationship indicates that SYFLOW's optimisation objective is highly sensitive to differences in performance distributions, and that such differences may remain small even when the discovered subgroup structure differs substantially.

Importantly, the observed KL divergence results further suggest that no sampling strategy consistently preserves performance distributions across all runs. Instead, for all evaluated strategies, there exist individual runs in which the sampled configuration set induces sufficiently strong distortions in the performance distribution to hinder SYFLOW's ability to discover the reference subgroups in respective samples. These cases do not appear to be tied to a specific sampling strategy, but rather occur sporadically across all stochastic approaches. This strongly indicates that distortions in the distribution of performance values are primarily driven by the specific sampled configurations, and is therefore highly dependent on the random seed or sample composition rather than the sampling strategy itself.

As a result, distortions in performance distributions within sample sets appear to be driven primarily by the specific random seed and resulting sampled configurations, rather than by the choice of sampling strategy itself. While stochastic strategies differ markedly in

how consistently they enable structural recovery, none of them reliably prevent occasional distortions of the performance distribution.

Answer to RQ1: Sampling Strategy Stochastic sampling strategies (random, distance-based, and diversified distance-based) tend to generate sampled configuration sets that enable SYFLOW to recover structurally similar subgroups more consistently. Among these, distance-based sampling offers a favourable balance between structural similarity and reduced variability across runs, making it a robust choice for subgroup discovery under sampling constraints.

Deterministic strategies, such as solver-based and t-wise sampling, typically result in a much smaller number of discovered subgroups. Within this limited set, SYFLOW is often able to recover individual reference subgroups with high structural similarity. However, this recovery is highly inconsistent across subgroups and datasets. Moreover, structurally well-matched subgroups do not reliably correspond to the most exceptional subgroups present in the full configuration space. Consequently, results obtained from deterministic strategies alone are less reliable and should be interpreted with caution. These strategies may therefore be better suited as complementary approaches, for example to validate or refine subgroups identified using stochastic sampling.

Meanwhile, distortions in performance distributions are observed across all evaluated sampling strategies and are not confined to any single approach. SYFLOW's subgroup discovery process is highly sensitive to such distortions, which can hinder the rediscovery of reference subgroups even when structural similarity would otherwise be expected.

4.2.2 RQ2: Sample Size

When analysing the effect of sample size on subgroup discovery, our results show that both the relative sampling budget and the resulting absolute number of sampled configurations play a crucial role in determining the stability and quality of discovered subgroups. While percentage-based sample sizes provide a convenient normalisation across datasets, Table 4.2 highlights that identical relative budgets can correspond to vastly different absolute sample sizes depending on the size of the overall configuration space.

At small sample sizes between 1% and 10%, subgroup discovery results exhibit high variability across most datasets, as reflected by wide interquartile ranges for precision and recall (Fig. 4.5 and Fig. 4.6). This instability is particularly pronounced for systems with small configuration spaces, such as LLVM and x264, where even 5% or 10% sampling corresponds to only a few dozen configurations. In these cases, minor changes in the sampled configurations can substantially alter the performance distribution observed by SYFLOW, leading to fluctuating subgroup discovery outcomes. In contrast, larger systems such as VP9 and JAVAGC already reach comparatively large absolute sample sizes at low relative budgets, allowing these datasets to stabilize at lower relative sample sizes.

As sample size increases, subgroup discovery becomes markedly more stable across all datasets. For most systems, variability decreases substantially once sample sizes reach approximately 20%. However, this threshold should not be interpreted as a universal rule. Rather, it represents a point at which the sampled configuration sets appear sufficiently

large to capture the dominant performance patterns of the underlying configuration space for the datasets studied. For small datasets, the absolute sample sizes corresponding to this threshold range from around 200 to 500 configurations. For large datasets, this stabilisation often occurs at 5% of their relative size, despite their absolute sample sizes already reaching close to a thousand configurations at 1%. This suggests that stable subgroup discovery relies on both absolute sample size, which enables reliable estimation of performance distributions and subgroup boundaries, and relative sample size, which ensures adequate coverage of the configuration space and reduces sampling bias.

Beyond these stabilisation points, increasing the sample size further yields diminishing returns. Median precision, recall, and F1 scores improve only marginally when increasing the sampling budget from 20% to 50%, independent of the absolute size of the dataset. This suggests that once a sufficient level of coverage has been achieved, additional sampled configurations contribute little new information for subgroup discovery, instead primarily reinforcing already discovered patterns.

A similar interpretation applies to performance-distribution divergence. As shown in Figure 4.8, increasing the sample size primarily reduces variance in the KL–F1 space, with points concentrating more tightly at higher F1 scores and lower KL divergence differences. In contrast, increasing the sample size does not introduce qualitatively new relationships between structural similarity and performance-distribution similarity. Instead, the same characteristic KL–F1 relationship observed for smaller samples persists across all sample sizes. This indicates that while larger samples improve the reliability of performance-distribution estimates, they do not fundamentally change SYFLOW's sensitivity to distributional distortions.

Taken together, these results indicate that the primary role of sample size lies in stabilising subgroup discovery by reducing sampling-induced noise and variability. Larger samples improve subgroup quality mainly by making subgroup recovery more reliable and less dependent on the specific sampled configurations. However, once the sampled data sufficiently captures the dominant performance patterns of the configuration space, further increases in sample size yield diminishing returns, both in terms of structural similarity and agreement of performance distributions.

Answer to RQ2: Sample Size Small sampling budgets frequently result in unstable subgroup discovery outcomes. Stability improves substantially once the sampled configuration set reaches both a sufficient absolute size and a sufficient relative size. In our study, an absolute sample size of approximately 200–500 configurations is typically required to reliably estimate performance distributions and subgroup boundaries, while a sufficiently large relative sample size is necessary to ensure adequate coverage diversity and avoid sampling bias. For moderately sized configuration spaces, these conditions often coincide at relative sampling budgets of around 20%, whereas for large configuration spaces stabilisation may already occur at relative budgets as low as 5%, once comparable absolute sample sizes are reached. Beyond this point, further increases in sample size yield diminishing returns, primarily reducing variance rather than improving subgroup quality.

Performance-distribution divergence follows a similar pattern, with larger samples reducing variability but not fundamentally altering the relationship between structural similarity and performance-distribution similarity. This suggests that sample size primarily affects

the robustness of subgroup discovery rather than the underlying optimisation behaviour of Syflow.

4.3 Threats to Validity

This section discusses potential threats to the validity of the presented results. We distinguish between threats to *internal validity*, which concern the correctness and reliability of the experimental setup and measurements, and threats to *external validity*, which concern the generalisability of the findings beyond the studied setting.

4.3.1 Internal Validity

A first threat to internal validity arises from our choice of hyperparameters for Syflow across all experiments. While these hyperparameters may not be optimal for every dataset, using a static setup ensures comparability across sampling strategies and datasets. By fixing the hyperparameters, we isolate the effects of sampling strategy and sample size, which is the primary focus of this study. Nevertheless, alternative parameterisations of Syflow could yield different subgroup structures or stability characteristics.

Second, the datasets used in this study rely on pre-existing performance measurements, which may contain noise due to uncontrolled environmental factors, system variability, or measurement imprecision. Nevertheless, while this noise cannot be fully eliminated, its impact is expected to affect all sampling strategies similarly and therefore primarily increases variance rather than systematically biasing comparisons.

Third, several evaluated sampling strategies are stochastic in nature, including random, distance-based, and diversified distance-based sampling. To reduce the influence of randomness, all stochastic experiments were repeated across multiple random seeds, and results are reported using distributional statistics rather than single runs. This allows us to explicitly analyse variability and stability as part of the evaluation.

Another potential threat stems from the subgroup matching procedure used during evaluation. Subgroups are aligned using a one-to-one matching based on maximal F1 score, which ensures interpretability and avoids assumptions about subgroup ordering. However, this matching does not guarantee a globally optimal assignment across all subgroups and ignores performance-distribution similarity when forming matches. While alternative matching strategies could yield slightly different aggregate similarity values, the consistent trends observed across datasets and sampling strategies suggest that the main conclusions are robust to these choices.

Finally, the KL divergence used for evaluation differs from the continuous KL divergence optimised internally by Syflow. Although the weighted formulation mirrors the structure of the optimisation objective, evaluation relies on a discrete approximation over empirical distributions. As a result, KL divergence values should be interpreted comparatively rather than as exact reflections of the optimisation objective. Despite this limitation, using a consistent divergence measure across all experiments ensures fair comparisons between sampling strategies.

4.3.2 External Validity

The primary threat to external validity is that this study exclusively evaluates SYFLOW as the subgroup discovery method. While this limits direct generalisation to other subgroup discovery algorithms, focusing on a single method allows us to clearly attribute observed effects to sampling strategy and sample size rather than algorithmic differences. Moreover, SYFLOW's emphasis on performance-distribution modelling makes it particularly suitable for studying sampling-induced distortions, which are central to this work. Still, future work should explore alternative subgroup discovery algorithms and hyperparameter setups to assess the generalisability of these findings.

Additionally, the set of sampling strategies considered, while representative, is not exhaustive. We include random, deterministic solver-based, combinatorial t-wise, and distance-based approaches, covering both stochastic and deterministic families. Although other sampling strategies or hybrid approaches may exhibit different behaviour, the selected strategies represent commonly used and conceptually distinct techniques, providing a broad basis for comparison.

The evaluation is further limited to eight configurable software systems. However, these systems span diverse domains, configuration space sizes, and complexity levels, ranging from small to very large configuration spaces and are established in prior literature [19]. This diversity supports a degree of generalisability, even though results may not directly extend to systems with extremely large, continuous, or highly non-Boolean configuration domains.

Finally, this thesis focuses exclusively on runtime performance as the target variable. Other non-functional properties, such as memory consumption or energy usage, may interact differently with sampling and subgroup discovery. Nevertheless, runtime performance is a widely studied and practically relevant metric [10, 11, 16, 20], making the findings applicable to a broad class of performance analysis scenarios.

5

Related Work

In this chapter, we review work closely related to the contributions of this thesis. We first discuss sampling strategies for configurable software systems and their applications and challenges in performance analysis. We then outline approaches based on performance-influence models and feature-interaction detection. Finally, we introduce subgroup discovery as a data-mining technique and discuss its relationship to performance analysis in configurable software systems.

Sampling Strategies A wide range of sampling strategies has been proposed to deal with the exponentially large configuration spaces of configurable software systems. Baseline approaches include uniform random sampling [19, 26] as well as solver-based sampling [6, 13], which relies on constraint solvers to generate valid configurations that respect feature-model constraints. In practice, solver-based approaches enumerate the first n solutions produced by the solver, which may bias the sample towards particular regions of the configuration space. To mitigate this effect, solver-based approaches are often combined with randomisation [5, 6, 13–15], or additional coverage criteria [17].

Coverage-oriented strategies are commonly based on combinatorial interaction testing, most notably t-wise sampling [17, 24, 26]. A t-wise sample ensures that all combinations of any t configuration options occur in at least one sampled configuration. Such strategies are effective at exposing low-order feature interactions but are limited to interactions of bounded size and do not necessarily yield a diverse representation of the configuration space, as coverage guarantees may be satisfied by relatively few and structurally similar configurations, particularly in the presence of sparse feature constraints [17, 26, 30].

Another line of work compiles feature models into decision-diagram representations (BDD/MDD) to enable efficient counting, reasoning, and, in some cases, uniform or distribution-aware sampling of valid configurations [4, 9]. Recent work in this direction demonstrates improved scalability and compactness of multi-valued decision diagrams for counting and representing large feature-model product sets [4], and other work proposes scalable samplers for extremely large configuration spaces such as the *Linux* kernel [9].

Distance-based sampling takes a complementary perspective by explicitly maximising diversity among sampled configurations. Instead of guaranteeing interaction coverage, these approaches aim to distribute configurations evenly across the configuration space according to a distance metric. Kaltenecker et al. [19] propose distance-based sampling schemes that ensure balanced coverage across different configuration distances, which has been shown to improve the representativeness of samples for performance analysis tasks. An extension of this approach, diversified distance-based sampling [19], further balances option frequencies to avoid over-representing common features.

Comparative studies highlight that these strategies trade off coverage, diversity, and measurement cost in different ways. Varshosaz et al. [30] classify sampling strategies along dimensions such as required input artefacts, sampling algorithm, and evaluation criteria, illustrating the lack of a universally optimal strategy. Similarly, Medeiros et al. [26] show that combining multiple sampling strategies can outperform individual approaches for fault detection, even when individual samplers perform well in isolation.

Performance-Influence Models and Interaction Detection Performance-influence models aim to learn how configuration options and their interactions affect non-functional properties such as runtime, latency, or throughput. Most approaches treat the system as a black box, measuring a subset of configurations, and a predictive model is trained to estimate performance for unseen configurations. Common modelling techniques include linear regression, decision trees, and ensemble methods, which balance predictive accuracy with interpretability [1, 28]. Tree-based models [10] are particularly popular in performance-influence modelling due to their ability to capture non-linear effects and feature interactions while remaining relatively interpretable. More recently, neural-network-based models [11] have been explored to capture complex, non-linear performance effects that are difficult to express with traditional learners.

Beyond prediction, several approaches explicitly target the detection of feature interactions. Siegmund et al. [28] define feature interactions as non-additive effects of option combinations and propose methods to automatically detect such interactions from measurement data. These techniques aim to identify small sets of interacting options that explain unexpected performance behaviour, a goal closely related to subgroup discovery. Complementary work studies not only how to detect interactions but also how to characterise and classify them. For example, Apel et al. [2] provide a systematic classification of feature interactions in configurable systems and discuss their implications for analysis and testing. Kolesnikov et al. [20] show that, across a range of real-world configurable systems, performance-influence models are dominated by low-order interactions involving only two or three options, which explains why relatively small and interpretable models can achieve high accuracy in practice. Dynamic, variational-execution-based approaches such as VarXplorer [29] provide lightweight processes and visualisations to expose which features interact and to help classify interactions as benign or suspicious in practice.

Recent work explores more expressive learners and transfer learning. DeepPerf, proposed by Ha and Zhang [11], uses deep neural networks to capture complex, non-linear interactions in large configuration spaces and has been shown to outperform traditional regression models in high-dimensional settings. Transfer-learning approaches reuse performance knowledge across environments, such as different hardware platforms or workloads, to reduce measurement cost. Jamshidi et al. [16] show that small environmental changes can often be handled via simple transformations, whereas larger changes require transferring structural knowledge, such as influential features, to guide new sampling.

White-box approaches complement black-box models by incorporating program analysis or profiling. For example, Weber et al. [31] combine lightweight profiling with learning to attribute performance effects to specific code regions, yielding more actionable explanations. While such methods are powerful, they require access to source code and instrumentation, and are therefore orthogonal to the black-box sampling scenarios studied in this thesis.

Subgroup Discovery Subgroup discovery originates from data mining and aims to identify interpretable descriptions of subpopulations whose behaviour deviates from that of the overall population. Formally, subgroups are defined by simple rules, such as conjunctions of attribute conditions, and are evaluated using quality measures that balance subgroup size and deviation from the global distribution [3, 12].

Early work on subgroup discovery focused on making the search feasible despite the exponential number of possible subgroup descriptions by employing efficient search strategies and pruning techniques that eliminate unpromising candidates early [3]. Later work extended these methods to numeric target variables and showed that carefully designed bounds can further reduce the search space [25]. Overall, these studies demonstrate that pruning strategies and suitable quality measures are essential for scalable and robust subgroup discovery [12].

Sampling has also been studied directly in the context of subgroup discovery. Scholz [27] proposes a knowledge-based sampling approach that reweights the data distribution based on previously discovered rules, biasing subsequent discovery towards novel and rare patterns. This iterative sampling scheme yields smaller and more diverse subgroup sets and illustrates how sampling choices can substantially influence both the stability and diversity of discovered subgroups.

Overall, subgroup discovery differs from performance-influence modelling in its focus on local, interpretable explanations rather than global predictive accuracy. At the same time, its goal of identifying small, exceptional regions of the configuration space closely aligns it with feature-interaction detection methods in configurable software systems.

6

Concluding Remarks

In this thesis, we conducted an empirical study on eight real-world configurable software systems to investigate how sampling strategies and sample size influence subgroup discovery outcomes with SYFLOW. For this purpose, we generated sample sets using several established sampling strategies and applied SYFLOW to both sampled and full datasets. We then compared the resulting subgroups to assess how sampling affects the ability of SYFLOW to discover the same subgroups previously identified on the full configuration spaces.

Our results show that sample sets generated by stochastic strategies (random, distance-based, diversified distance-based) often allow SYFLOW to discover subgroups that closely resemble reference subgroups in terms of structure, although the consistency of this recovery varies across strategies and datasets. Among these, distance-based sampling frequently yields a favourable balance between structural similarity and result stability, whereas random sampling, while capable of producing highly similar subgroups, suffers from substantial variability across runs. Deterministic strategies, i.e., our solver-based implementation and t-wise sampling, typically result in a much smaller set of recovered subgroups. Within these limited results, individual reference subgroups may be recovered well, but recovery quality varies strongly across datasets and subgroups, which limits the reliability of these approaches when used in isolation.

Meanwhile, sampling can alter the observed performance behaviour of subgroups under all evaluated strategies. Across datasets, subgroups discovered from sampled data may still exhibit performance distributions similar to those of reference subgroups even when their structural similarity is low, and vice versa. This indicates that stochastic factors, namely the random seed and sampled configurations, more strongly influence performance distributions than sampling strategies or different sample sizes do. Further, it suggests that SYFLOW's subgroup discovery process remains sensitive to small changes in the sampled performance data, and similarity in performance distributions alone does not reliably imply structural agreement.

In further experiments, we find that very small sample sizes lead to highly unstable subgroup discovery outcomes, characterised by large variability in structural similarity and frequent mismatches to reference subgroups. As sample size increases, subgroup discovery becomes more stable, allowing subgroups from the sample set to match reference subgroups more consistently across systems. However, these improvements plateau once moderate sample sizes are reached, with further increases yielding diminishing returns. Notably, while larger sample sizes reduce variance in performance distributions, the structural agreement between discovered subgroups and reference subgroups remains largely unchanged compared to the observations in smaller sample sizes.

In conclusion, we show that both sampling strategy and sample size substantially influence the reliability of subgroup discovery in configurable software systems. Sampled configuration sets that provide sufficient coverage and stability, particularly those obtained via distance-based sampling at sample sizes of around 20% and an absolute size of at least 200 configurations, offer the most reliable conditions for recovering meaningful subgroup structure. At the same time, sampling-induced distortions of performance distributions can occur regardless of strategy or sample size, highlighting the importance of repeated sampling, robustness checks, and cautious interpretation of subgroup discovery results.

A

Appendix

In this chapter, we provide additional results collected in our experimental study. We begin by reporting an analysis of discovery distributions and discovery progression, which quantifies how many high-quality reference subgroups are recovered across runs for each stochastic sampling strategy. These results were used to support qualitative observations in the main evaluation regarding the consistency and multiplicity of subgroup recovery.

We then present results for distance-based and diversified distance-based sampling under increasing sample sizes, which were omitted from the main evaluation due to technical issues during data collection. Finally, we report results of a Spearman rank correlation analysis between structural similarity and performance-distribution similarity for deterministic sampling strategies.

A.1 Discovery Distribution and Progression

This section reports supplementary statistics on how many high-quality reference subgroups are recovered across runs for each stochastic sampling strategy.

Dataset	n=0	n=1	n=2	n=3	n=4	n=5+	Total	Mean
7z	2.2%	22.2%	55.4%	19.2%	1.0%	0.0%	500	1.95
BerkeleyDBC	6.6%	39.2%	36.4%	14.6%	2.2%	1.0%	500	1.70
Dune	46.4%	40.1%	12.6%	0.8%	0.0%	0.0%	491	0.68
JavaGC	2.4%	9.0%	46.8%	36.0%	5.8%	0.0%	500	2.34
LLVM	27.8%	44.0%	25.1%	2.9%	0.2%	0.0%	414	1.04
Polly	17.3%	21.9%	48.3%	12.1%	0.4%	0.0%	462	1.56
VP9	38.5%	27.6%	20.5%	12.3%	1.0%	0.0%	478	1.10
x264	55.8%	34.6%	8.7%	0.6%	0.2%	0.0%	471	0.55
All	**24.3%**	**29.5%**	**32.0%**	**12.7%**	**1.4%**	**0.1%**	**3816**	**1.38**

Table A.1: Discovery distribution for *random* sampling.
Each entry reports the percentage of runs in which SYFLOW recovered exactly n reference subgroups with an F1 score of at least 0.9. The *mean* column denotes the average number of such high-quality subgroup matches per run across all runs for the dataset.

Dataset	0→1	1→2	2→3	3→4	4→5	Runs
7z	97.8%	77.3%	26.7%	5.0%	0.0%	500
BerkeleyDBC	93.4%	58.0%	32.8%	18.0%	31.2%	500
Dune	53.6%	25.1%	6.1%	0.0%	—	491
JavaGC	97.6%	90.8%	47.2%	13.9%	0.0%	500
LLVM	72.2%	39.1%	11.1%	7.7%	0.0%	414
Polly	82.7%	73.6%	20.6%	3.4%	0.0%	462
VP9	61.5%	55.1%	39.5%	7.8%	0.0%	478
x264	44.2%	21.6%	8.9%	25.0%	0.0%	471
All	**75.7%**	**61.0%**	**30.7%**	**10.9%**	**8.5%**	**3816**

Table A.2: Discovery progression for *random* sampling.
Each column $n \to n{+}1$ reports the conditional probability (in percent) that a run which recovered n reference subgroups with F1 ≥ 0.9 also recovered at least $n{+}1$ such subgroups.

Dataset	n=0	n=1	n=2	n=3	n=4	n=5+	Total	Mean
7z	9.0%	66.2%	20.2%	4.4%	0.2%	0.0%	500	1.21
BerkeleyDBC	5.0%	25.0%	45.8%	23.8%	0.4%	0.0%	500	1.90
Dune	0.0%	0.0%	50.0%	50.0%	0.0%	0.0%	200	2.50
JavaGC	7.0%	39.2%	46.8%	7.0%	0.0%	0.0%	500	1.54
LLVM	12.6%	53.9%	29.9%	3.6%	0.0%	0.0%	388	1.24
Polly	23.6%	47.1%	26.5%	2.7%	0.0%	0.0%	437	1.08
VP9	62.0%	25.0%	10.7%	2.3%	0.0%	0.0%	476	0.53
x264	42.2%	46.7%	11.1%	0.0%	0.0%	0.0%	415	0.69
All	**21.3%**	**40.4%**	**29.1%**	**9.2%**	**0.1%**	**0.0%**	**3416**	**1.26**

Table A.3: Discovery distribution for *distance-based* sampling.
Each entry reports the percentage of runs in which SYFLOW recovered exactly n reference subgroups with an F1 score of at least 0.9. The *mean* column denotes the average number of such high-quality subgroup matches per run across all runs for the dataset.

Dataset	0→1	1→2	2→3	3→4	4→5	Runs
7z	91.0%	27.3%	18.5%	4.3%	0.0%	500
BerkeleyDBC	95.0%	73.7%	34.6%	1.7%	0.0%	500
Dune	100.0%	100.0%	50.0%	0.0%	—	200
JavaGC	93.0%	57.8%	13.0%	0.0%	—	500
LLVM	87.4%	38.3%	10.8%	0.0%	—	388
Polly	76.4%	38.3%	9.4%	0.0%	—	437
VP9	38.0%	34.3%	17.7%	0.0%	—	476
x264	57.8%	19.2%	0.0%	—	—	415
All	**78.7%**	**48.7%**	**24.1%**	**0.9%**	**0.0%**	**3416**

Table A.4: Discovery progression for *distance-based* sampling.
Each column $n \to n{+}1$ reports the conditional probability (in percent) that a run which recovered n reference subgroups with F1 ≥ 0.9 also recovered at least $n{+}1$ such subgroups.

Dataset	n=0	n=1	n=2	n=3	n=4	n=5+	Total	Mean
7z	9.4%	40.6%	41.4%	8.2%	0.4%	0.0%	500	1.50
BerkeleyDBC	6.6%	38.8%	42.2%	10.2%	2.2%	0.0%	500	1.63
Dune	0.5%	74.0%	9.9%	15.6%	0.0%	0.0%	192	1.41
JavaGC	11.8%	37.6%	39.6%	9.2%	1.8%	0.0%	500	1.52
LLVM	24.6%	47.6%	23.7%	4.1%	0.0%	0.0%	418	1.07
Polly	47.2%	41.4%	11.4%	0.0%	0.0%	0.0%	394	0.64
VP9	58.5%	27.7%	10.1%	3.7%	0.0%	0.0%	487	0.59
x264	32.6%	48.1%	18.7%	0.6%	0.0%	0.0%	466	0.87
All	**25.1%**	**41.9%**	**26.5%**	**6.0%**	**0.6%**	**0.0%**	**3457**	**1.15**

Table A.5: Discovery distribution for *diversified distance-based* sampling.
Each entry reports the percentage of runs in which SYFLOW recovered exactly n reference subgroups with an F1 score of at least 0.9. The *mean* column denotes the average number of such high-quality subgroup matches per run across all runs for the dataset.

Dataset	0→1	1→2	2→3	3→4	4→5	Runs
7z	90.6%	55.2%	17.2%	4.7%	0.0%	500
BerkeleyDBC	93.4%	58.5%	22.7%	17.7%	0.0%	500
Dune	99.5%	25.7%	61.2%	0.0%	—	192
JavaGC	88.2%	57.4%	21.7%	16.4%	0.0%	500
LLVM	75.4%	36.8%	14.7%	0.0%	—	418
Polly	52.8%	21.6%	0.0%	—	—	394
VP9	41.5%	33.2%	26.9%	0.0%	—	487
x264	67.4%	28.7%	3.3%	0.0%	—	466
All	**74.9%**	**44.1%**	**19.9%**	**9.6%**	**0.0%**	**3457**

Table A.6: Discovery progression for *diversified distance-based* sampling.
Each column $n \to n{+}1$ reports the conditional probability (in percent) that a run which recovered n reference subgroups with F1 ≥ 0.9 also recovered at least $n{+}1$ such subgroups.

A.2 Distance-Based Sampling with Increasing Sample Sizes

In this section, we present results for distance-based and diversified distance-based sampling strategies evaluated under increasing sample sizes. We exclude results for *Dune* due to technical issues during data collection.

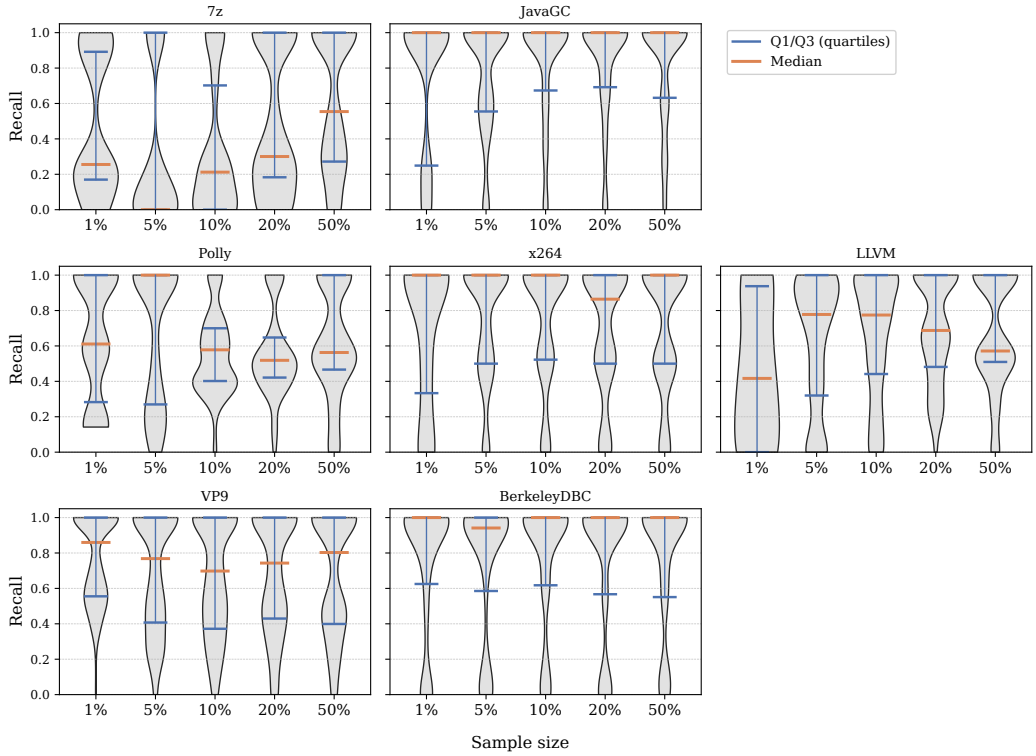

Figure A.1: Recall distributions across sample sizes for distance-based sampling.

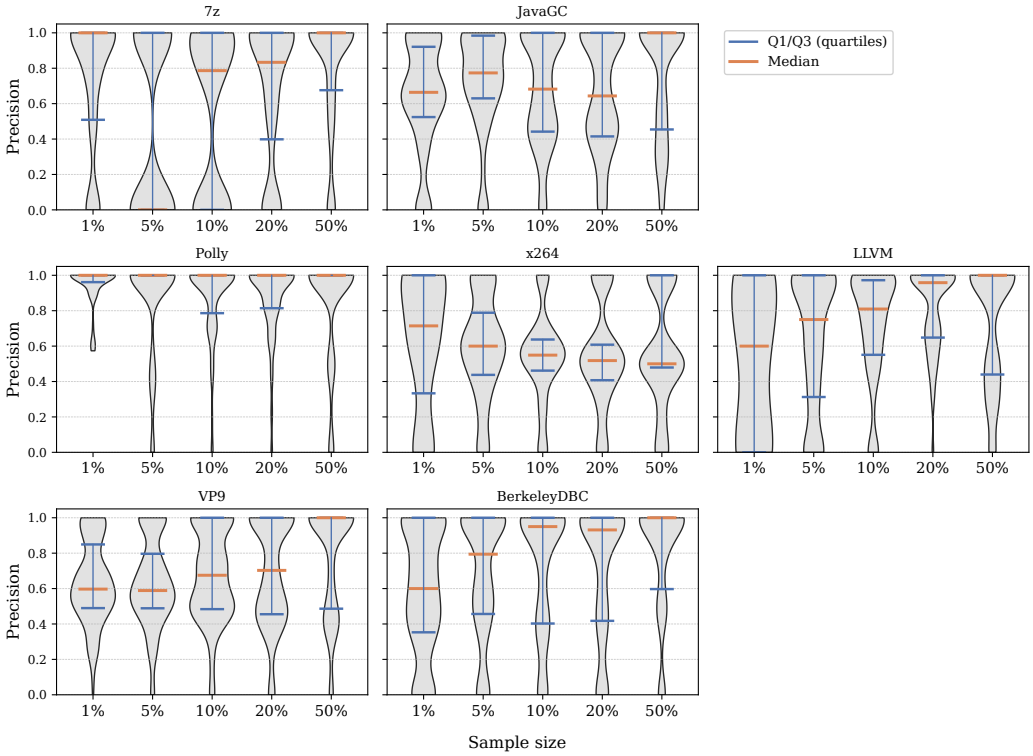

Figure A.2: Precision distributions across sample sizes for distance-based sampling.

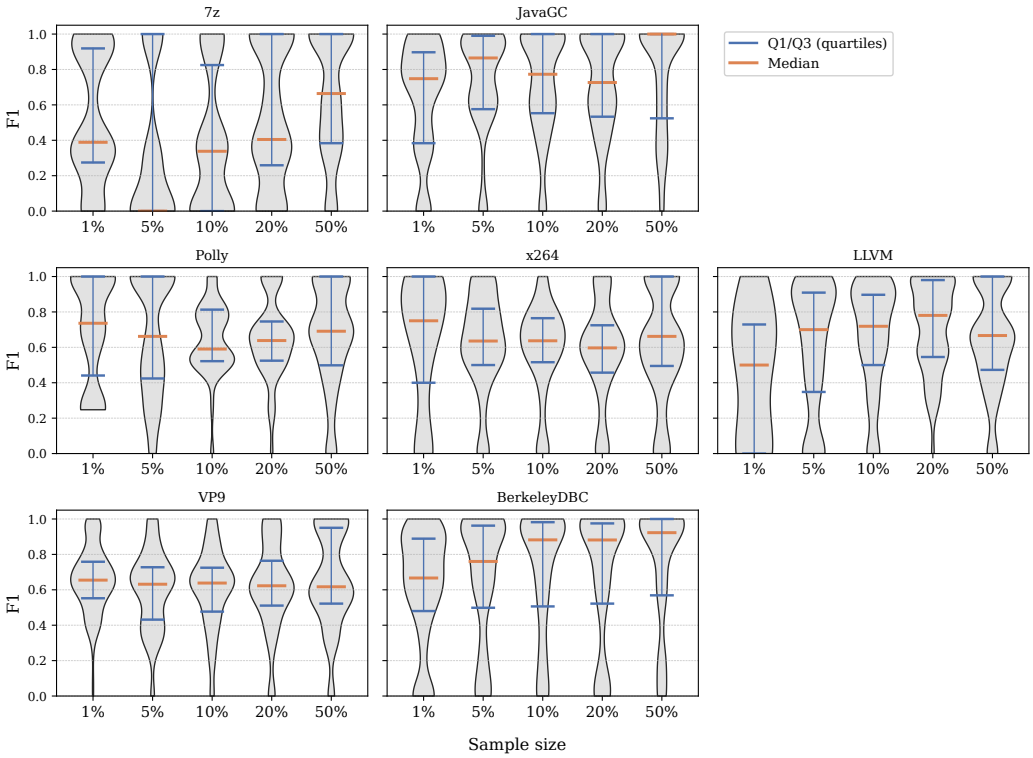

Figure A.3: F1-score distributions across sample sizes for distance-based sampling.

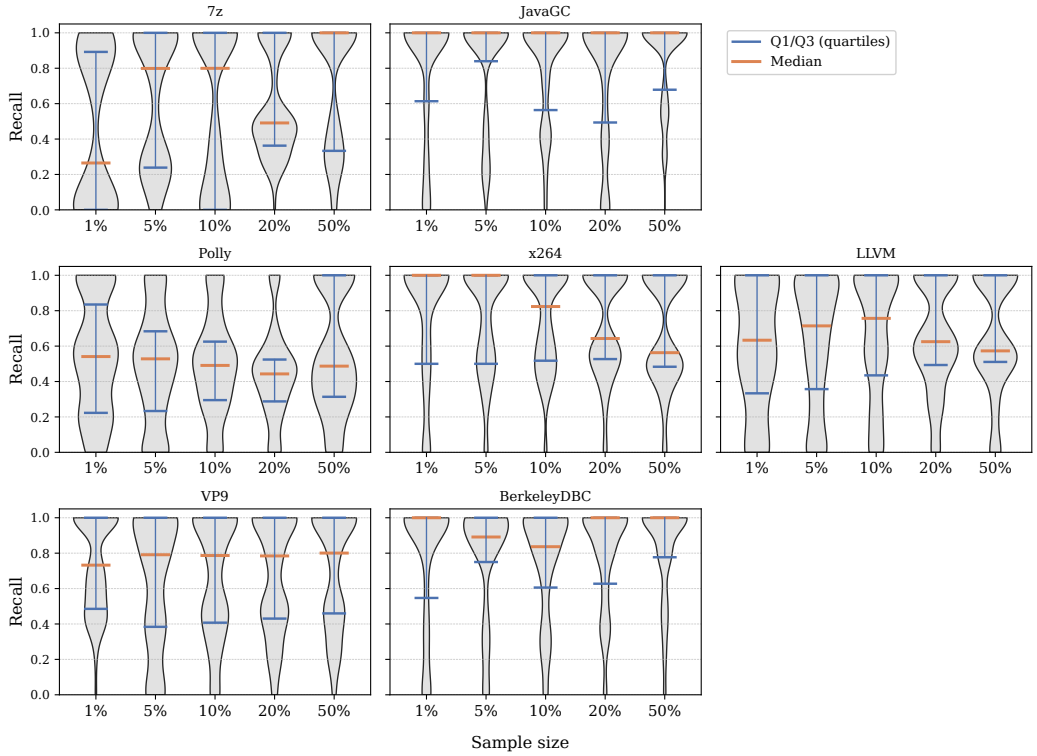

Figure A.4: Recall distributions across sample sizes for diversified distance-based sampling.

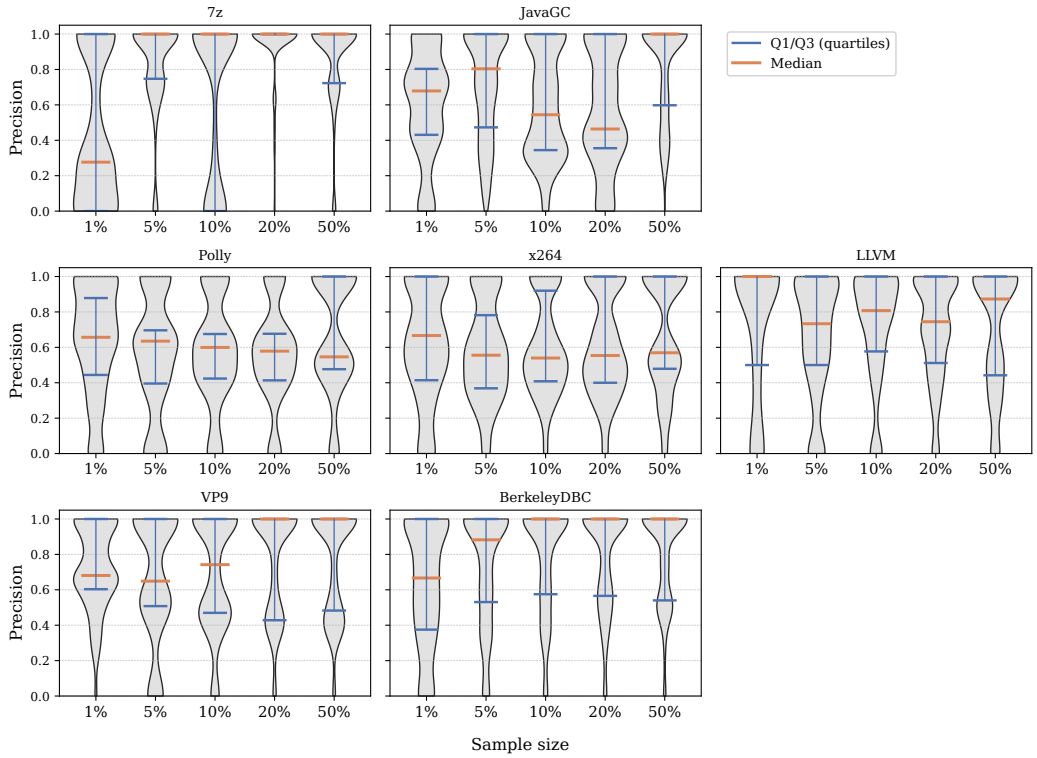

Figure A.5: Precision distributions across sample sizes for diversified distance-based sampling.

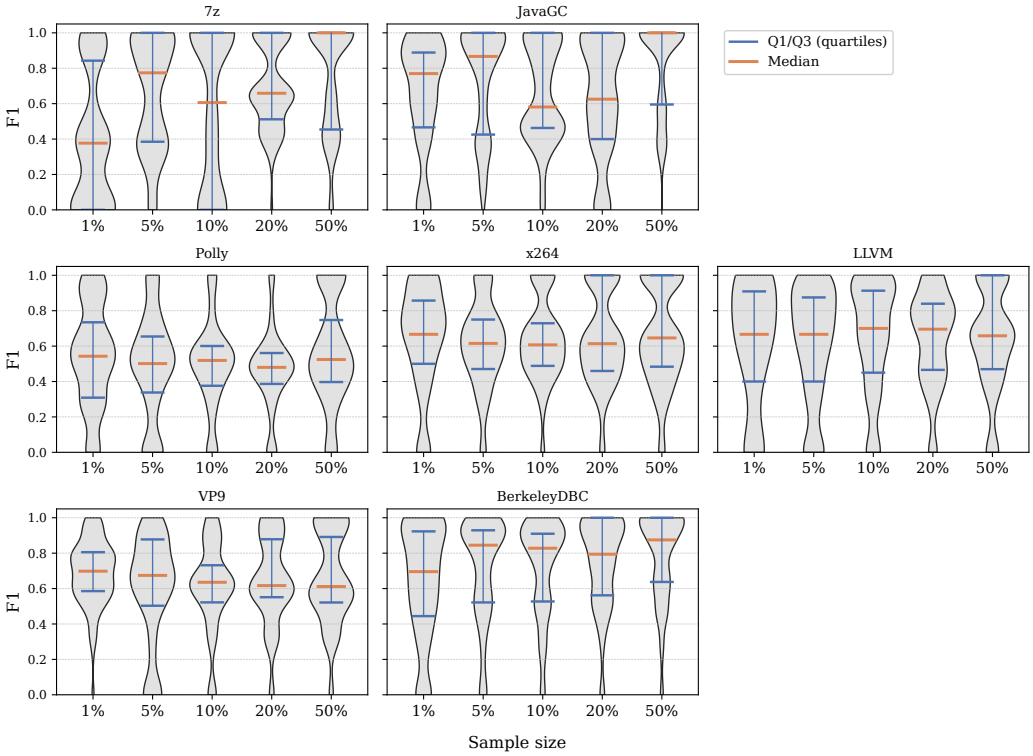

Figure A.6: F1-score distributions across sample sizes for diversified distance-based sampling.

Statement on the Usage of Generative Digital Assistants

For this proposal, the following generative digital assistants have been used:
We have used *ChatGPT*[1] for text revision to address grammatical errors and ensure formal phrasing.
We have used *GitHub Copilot*[2] for code completion and general help with *Python*'s syntax (for example, to create the intersection of two sets). The tool was further used to refactor and simplify existing code to aid readability and resolve issues. Suggestions by it have been verified by reading up on *Python*'s or the respective library's documentation and manual debugging.
We are aware of the potential dangers of using these tools and have used them sensibly with caution and with critical thinking.

1 https://chat.openai.com/
2 https://github.com/features/copilot

Bibliography

[1] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. *Feature-oriented software product lines*. 2013.

[2] Sven Apel, Sergiy Kolesnikov, Norbert Siegmund, Christian Kästner, and Brady Garvin. "Exploring feature interactions in the wild: the new feature-interaction challenge." In: *Proceedings of the 5th international workshop on feature-oriented software development*. 2013, pp. 1–8.

[3] Martin Atzmueller. "Subgroup discovery." In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5.1 (2015), pp. 35–49.

[4] Andrea Bombarda and Angelo Gargantini. "On the Use of Multi-valued Decision Diagrams to Count Valid Configurations of Feature Models." In: *Proceedings of the 28th ACM International Systems and Software Product Line Conference*. 2024, pp. 96–106.

[5] Supratik Chakraborty, Daniel Fremont, Kuldeep Meel, Sanjit Seshia, and Moshe Vardi. "Distribution-aware sampling and weighted model counting for SAT." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1. 2014.

[6] Supratik Chakraborty, Kuldeep S Meel, and Moshe Y Vardi. "A scalable and nearly uniform generator of SAT witnesses." In: *International Conference on Computer Aided Verification*. 2013, pp. 608–623.

[7] Thomas M Cover. *Elements of information theory*. 1999.

[8] Hercules Dalianis. "Evaluation metrics and evaluation." In: *Clinical Text Mining: secondary use of electronic patient records*. 2018, pp. 45–53.

[9] David Fernandez-Amoros, Ruben Heradio, Christoph Mayr-Dorn, and Alexander Egyed. "Scalable sampling of highly-configurable systems: Generating random instances of the Linux kernel." In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 2022, pp. 1–12.

[10] Jianmei Guo, Krzysztof Czarnecki, Sven Apel, Norbert Siegmund, and Andrzej Wasowski. "Variability-aware performance prediction: A statistical learning approach." In: *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*. 2013, pp. 301–311.

[11] Huong Ha and Hongyu Zhang. "Deepperf: performance prediction for configurable software with deep sparse neural network." In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 2019, pp. 1095–1106.

[12] Sumyea Helal. "Subgroup discovery algorithms: a survey and empirical evaluation." In: *Journal of computer science and technology* 31.3 (2016), pp. 561–576.

[13] Christopher Henard, Mike Papadakis, Mark Harman, and Yves Le Traon. "Combining multi-objective search and constraint solving for configuring large software product lines." In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. 2015, pp. 517–528.

[14] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, and Yves Le Traon. "Towards automated testing and fixing of re-engineered feature models." In: *2013 35th International Conference on Software Engineering (ICSE)*. 2013, pp. 1245–1248.

[15] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, and Yves Le Traon. "Multi-objective test generation for software product lines." In: *Proceedings of the 17th International Software Product Line Conference*. 2013, pp. 62–71.

[16] Pooyan Jamshidi, Norbert Siegmund, Miguel Velez, Christian Kästner, Akshay Patel, and Yuvraj Agarwal. "Transfer learning for performance modeling of configurable systems: An exploratory analysis." In: *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2017, pp. 497–508.

[17] Martin Fagereng Johansen, Øystein Haugen, and Franck Fleurey. "An algorithm for generating t-wise covering arrays from large feature models." In: *Proceedings of the 16th International Software Product Line Conference-Volume 1*. 2012, pp. 46–55.

[18] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, and Sven Apel. "The Interplay of Sampling and Machine Learning for Software Performance Prediction." In: *IEEE Software* 37.4 (2020), pp. 58–66.

[19] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, Jianmei Guo, and Sven Apel. "Distance-Based Sampling of Software Configuration Spaces." In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 2019, pp. 1084–1094.

[20] Sergiy Kolesnikov, Norbert Siegmund, Christian Kästner, and Sven Apel. "Tradeoffs in modeling performance of highly configurable software systems." In: *Software and Systems Modeling* 18.3 (2019), pp. 2265–2283.

[21] D.R. Kuhn, D.R. Wallace, and A.M. Gallo. "Software fault interactions and implications for software testing." In: *IEEE Transactions on Software Engineering* 30.6 (2004), pp. 418–421.

[22] Solomon Kullback and Richard A Leibler. "On information and sufficiency." In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[23] Nada Lavrač, Branko Kavšek, Peter Flach, and Ljupčo Todorovski. "Subgroup discovery with CN2-SD." In: *Journal of Machine Learning Research* 5.Feb (2004), pp. 153–188.

[24] Yu Lei, Raghu Kacker, D. Richard Kuhn, Vadim Okun, and James Lawrence. "IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing." In: *Software Testing, Verification and Reliability* 18.3 (2008), pp. 125–148.

[25] Florian Lemmerich, Martin Atzmueller, and Frank Puppe. "Fast exhaustive subgroup discovery with numerical target concepts." In: *Data Mining and Knowledge Discovery* 30.3 (2016), pp. 711–762.

[26] 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*. 2016, pp. 643–654.

[27] Martin Scholz. "Knowledge-Based Sampling for Subgroup Discovery." In: *Local Pattern Detection*. 2005, pp. 171–189.

[28] 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*. 2015, pp. 284–294.

[29] Larissa Rocha Soares, Jens Meinicke, Sarah Nadi, Christian Kästner, and Eduardo Santana de Almeida. "Exploring feature interactions without specifications: A controlled experiment." In: *ACM SIGPLAN Notices* 53.9 (2018), pp. 40–52.

[30] 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*. 2018, pp. 1–13.

[31] Max Weber, Sven Apel, and Norbert Siegmund. "White-box performance-influence models: A profiling and learning approach." In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 2021, pp. 1059–1071.

[32] Sascha Xu, Nils Philipp Walter, Janis Kalofolias, and Jilles Vreeken. "Learning exceptional subgroups by end-to-end maximizing KL-divergence." In: *arXiv preprint arXiv:2402.12930* (2024).

[33] 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*. 2015, pp. 307–319.