Small Model Portfolios for Many Deployment Profiles: Submodular Coverage under Bundled Constraints

Ji Cheng
City University of Hong Kong
jicheng9617@gmail.com

Abstract

Deploying deep learning models across heterogeneous hardware requires each model to jointly satisfy a bundle of coupled constraints on accuracy, latency, memory, and energy. Maintaining a specialist for every deployment scenario is operationally prohibitive, so practitioners need a small portfolio of K models that collectively covers many deployment profiles. Existing hardware-aware NAS and Pareto-based approaches target marginal objectives and do not directly address this downstream decision. We formalize it as Deployment Portfolio Selection over a fixed pre-measured archive, where a profile is covered only when a single retained model simultaneously satisfies its entire constraint bundle. The induced objective is weighted maximum coverage, which is NP-hard but monotone submodular, so greedy selection inherits a $(1 - 1/e)$ guarantee, and sample-then-optimize admits a finite-sample bound. Across an 18-family sweep on HW-NAS-Bench, direct bundled-coverage optimization is most valuable in the small-portfolio, low-coverability regime: gains over fixed scalarization concentrate at $K = 1$–3 and on the hardest profiles, and persist under three non-uniform demand models, whereas scalarization tuned on additional validation profiles nearly matches Greedy under uniform demand. Complementary analyses of remeasurement noise, runtime, and compact feasibility storage clarify when bundled coverage is preferable to scalarized or Pareto-based alternatives.

1 Introduction

Modern deep learning models are increasingly deployed across a heterogeneous range of hardware platforms, from cloud GPUs for batch inference to edge TPUs in autonomous vehicles and microcontrollers in IoT sensors [30, 4, 26, 5, 17, 18, 28]. In such settings, each deployment scenario is governed not by a single requirement but by a bundle of coupled constraints: a model must satisfy a minimum accuracy threshold, a maximum inference latency budget, a peak memory budget, and sometimes an energy budget, all using the same checkpoint. We call such a bundle a **deployment profile**.

The prevailing paradigm handles these requirements only indirectly. Hardware-aware neural architecture search (HW-NAS) often targets one device or one dominant hardware constraint at a time, while multi-objective NAS methods approximate Pareto fronts over accuracy and hardware costs [20, 21]. These approaches are useful for exposing trade-offs, but they do not directly solve the downstream deployment decision considered here. First, Pareto optimality is defined over attribute vectors rather than bundled feasibility. A model that is Pareto-optimal in the accuracy–latency plane may still violate the memory or energy requirements of a concrete deployment profile, and is therefore unusable for that profile. Second, maintaining many checkpoints is operationally expensive because each additional model incurs storage, testing, monitoring, and update costs. In practice, organizations often prefer a small portfolio of models that can collectively handle a broad range of deployment scenarios.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

This paper introduces and studies the **Deployment Portfolio Selection** problem. The input consists of a fixed archive $\mathcal{M}$ of N candidate models, each measured on task-quality and hardware attributes, together with either a finite set or a distribution of deployment profiles and a budget K [15, 12]. The goal is to select up to K models from $\mathcal{M}$ so as to maximize the number, or expected fraction, of deployment profiles that are covered. A profile is counted as covered only when at least one selected model satisfies all constraints in the bundle simultaneously.

We make the following contributions:

- We formulate deployment portfolio selection over a fixed pre-measured model archive, where a deployment profile is counted as covered only if one retained model satisfies every constraint in the bundle simultaneously.
- Once deployment profiles are materialized, the objective becomes weighted maximum coverage. Consequently, NP-hardness, monotone submodularity, and the greedy $(1 - 1/e)$ approximation guarantee follow from classical coverage theory.
- Across an 18-family sweep on HW-NAS-Bench, we characterize when direct bundled coverage matters most, namely in the small-portfolio, low-coverability regime before saturation ($K = 5$ under uniform CIFAR-10 demand).
- We report weighted-demand experiments, baseline audits, normalization-sensitivity analyses, Pareto-front-restricted diagnostics, reduced-instance exact-gap checks, noise-robustness tests, and runtime/storage scaling analyses to clarify the operational regime in which bundled coverage is preferable to scalarized or Pareto-post-selection alternatives.

2 Related Work

Hardware-aware NAS and efficient deployment. Hardware-aware NAS [1] optimizes or specializes architectures under device-specific constraints such as latency, energy, and memory, as in FBNet [30], ProxylessNAS [4], MnasNet [26], and Once-for-All [5]. Benchmarks such as HW-NAS-Bench [15] and NATS-Bench [12] enable cross-device studies over a shared measured archive, while TinyML and deployment-systems work—including MCUNet [17], MCUNetV2 [18], and HTVM [28]—examines efficient execution on heterogeneous resource-constrained hardware. These works motivate our setting but target a different problem: searching for, specializing, or executing models for particular hardware. They do not address the downstream decision we study: given a fixed measured archive and a limited checkpoint budget, which small subset should be retained to serve a *distribution* of future deployment requirements?

Multi-objective search and Pareto fronts. When multiple metrics matter, a natural approach is to approximate the Pareto front over accuracy and hardware cost, as in multi-objective NAS [20, 21, 7, 10, 2, 3]. Pareto fronts expose trade-offs but do not directly solve our problem: Pareto optimality is defined over model-level objectives, whereas deployment requests arrive as bundled profiles whose constraints must be satisfied *jointly* by a single model. Even given a Pareto front, choosing a few checkpoints to maintain remains a representative-subset selection problem. Few-for-Many [19] moves closer to this view by seeking small complementary solution sets for many objectives; our setting differs in that profiles are concrete deployment bundles rather than abstract objectives, and a profile is covered only when one retained model simultaneously satisfies every constraint in its bundle.

Coverage-oriented formulations and submodular structure. Our formulation connects to coverage-oriented views of multi-criteria decision making that emphasize satisfactory regions rather than dense Pareto approximation [22], though the connection is primarily conceptual: we work with a discrete, pre-measured offline archive rather than a sequential experimental-design problem over a continuous space. Once profiles are materialized, the objective becomes an instance of weighted maximum coverage and falls within classical monotone submodular maximization under a cardinality constraint [23, 14]. This differs from covering multiple submodular constraints [6]: here, bundled deployment feasibility itself *induces* the coverage objective, and the decision variable is a small portfolio drawn from the archive.

Representative-subset objectives beyond hard coverage. Facility-location-style and diversity-aware subset selection seek small subsets that summarize a candidate pool or preserve broad coverage

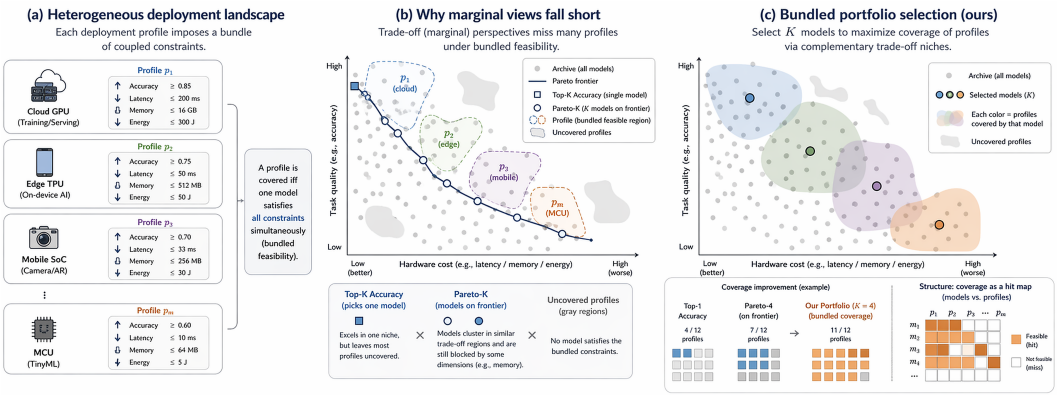


Figure 1: **Deployment portfolio selection under bundled feasibility.** A deployment profile is covered only when one model simultaneously satisfies its accuracy, latency, memory, and energy constraints. Instead of relying on marginal Pareto views, we formulate archive selection as weighted maximum coverage over the model–profile feasibility relation, selecting K complementary models to maximize covered profiles. The objective is monotone submodular and admits a greedy $(1 - 1/e)$ guarantee.

of a data space [16, 29]. Such objectives are effective when profiles admit meaningful notions of similarity or redundancy. Our objective is instead hard bundled feasibility, a profile contributes value only when a single retained model satisfies *all* thresholds in its bundle, so we view representative-subset objectives as complementary alternatives rather than direct surrogates.

Robust and probabilistic coverage under uncertainty. In practice, hardware costs and accuracy estimates fluctuate across measurement conditions, firmware versions, thermal states, and repeated runs. This motivates uncertainty-aware variants of submodular coverage, including distributionally robust and chance-constrained formulations [25, 9]. Our main formulation studies deterministic archive selection; Appendix G evaluates empirical sensitivity to attribute perturbations, and we leave full uncertainty-aware optimization to future work.

Positioning. We identify heterogeneous deployment with bundled feasibility as a concrete archive-selection problem and cast it as weighted profile coverage under a portfolio budget. The perspective is operational rather than predictive: unlike classical model selection [8] or ensembling, different deployment scenarios are served by different retained models, and the value of maintaining several models comes from *complementary coverage* rather than prediction aggregation. At deployment, a router can dispatch each covered profile to a feasible retained model. Early-exit networks [27] and mixtures of experts [24] can enter the archive at measured operating points. Joint execution requires measurements of the combined configuration. Appendix A.6 discusses the connection to fairness.

3 Problem Formulation

3.1 Setup and Notation

Let $\mathcal{M} = \{x_1, \ldots, x_N\}$ denote a finite **model archive** of N candidate models. Each model $x \in \mathcal{M}$ is characterized by a vector of d measurable attributes:

$$\boldsymbol{a}(x) = \big(a_1(x),\, a_2(x),\, \ldots,\, a_d(x)\big) \in \mathbb{R}^d, \tag{1}$$

where the attributes include task performance metrics (e.g., top-1 accuracy, mAP) and hardware cost metrics (e.g., latency on device h, peak memory, energy consumption). We assume that all attributes are measured or estimated for every model in the archive—this is the setting provided by NAS benchmarks such as HW-NAS-Bench and NATS-Bench.

Definition 1 (Deployment Profile)**.** *A **deployment profile** is a tuple $r = (S_r, \boldsymbol{\tau}_r, \boldsymbol{\sigma}_r)$, where:*

- $S_r \subseteq [d]$ *is a subset of* active attributes *relevant to this profile;*

- $\boldsymbol{\tau}_r = (\tau_{r,j})_{j \in S_r}$ *specifies a threshold for each active attribute;*
- $\boldsymbol{\sigma}_r = (\sigma_{r,j})_{j \in S_r} \in \{+1, -1\}^{|S_r|}$ *specifies the direction of each constraint (*$+1$*: attribute must be* $\geq \tau_{r,j}$*;* -1*: attribute must be* $\leq \tau_{r,j}$*).*

Definition 2 (Profile Satisfaction)**.** *A model* $x \in \mathcal{M}$ ***satisfies*** *a deployment profile* r*, written* $x \models r$*, if*

$$\sigma_{r,j} \cdot a_j(x) \;\geq\; \sigma_{r,j} \cdot \tau_{r,j}, \quad \forall\, j \in S_r. \tag{2}$$

Definition 3 (Profile Coverage)**.** *A portfolio* $A \subseteq \mathcal{M}$ ***covers*** *a profile* r *if there exists at least one model in* A *that satisfies* r*:*

$$\mathit{cov}(A, r) \;=\; \mathbf{1}\big[\exists\, x \in A : x \models r\big]. \tag{3}$$

Note the critical *bundled* nature of coverage: a profile is covered only if some *single* model in A meets *all* constraints in the bundle simultaneously. It is *not* sufficient for one model to meet the accuracy constraint and a different model to meet the latency constraint.

3.2 The Deployment Portfolio Selection Problem

We consider two problem variants depending on whether the set of deployment profiles is finite and known, or drawn from a distribution.

Finite-profile version. Let $\mathcal{R} = \{r_1, \ldots, r_M\}$ be a finite collection of deployment profiles with associated weights $w_1, \ldots, w_M \geq 0$ (reflecting, e.g., the frequency or business importance of each profile). The **Deployment Portfolio Selection** problem is:

$$\max_{A \subseteq \mathcal{M},\, |A| \leq K} \quad F(A) \;=\; \sum_{m=1}^{M} w_m \cdot \mathrm{cov}(A, r_m). \tag{4}$$

Theorem 1 (NP-Hardness)**.** *The Deployment Portfolio Selection problem* (4) *is NP-hard.*

This hardness result has a positive flip side: it tells us that the $(1 - 1/e)$-approximation achievable by the greedy algorithm is essentially the best we can hope for in polynomial time.

Distributional version. Let P be a distribution over deployment profiles. The distributional objective is:

$$\max_{A \subseteq \mathcal{M},\, |A| \leq K} \quad F(A) \;=\; \mathbb{E}_{r \sim P}\big[\mathrm{cov}(A, r)\big] \;=\; \mathbb{P}_{r \sim P}\big[\exists\, x \in A : x \models r\big]. \tag{5}$$

In practice, we observe n i.i.d. samples $r_1, \ldots, r_n \sim P$ and optimize the empirical objective $\hat{F}(A) = \frac{1}{n} \sum_{i=1}^{n} \mathrm{cov}(A, r_i)$.

Each deployment profile r induces a feasibility set $e_r = \{x \in \mathcal{M} : x \models r\}$, so the finite-profile objective in (4) is exactly a weighted maximum-coverage objective over $\{e_r\}_{r \in \mathcal{R}}$. We defer the hypergraph interpretation and its relation to composite reductions to Appendix A.1.

Scope and assumptions. The formulation assumes a fixed pre-measured archive and deterministic attribute values at selection time. In the distributional setting, the sampled profiles are assumed i.i.d. from P and independent of the archive. These assumptions match the offline archive-selection setting provided by benchmarks such as HW-NAS-Bench and NATS-Bench, but they do not by themselves model remeasurement noise, attribute-estimation uncertainty, or profile shift. We therefore interpret the generalization result below as a finite-sample guarantee for the sample-then-optimize procedure within this offline setting.

4 Algorithm

4.1 Greedy Portfolio Selection

Since the coverage objective $F(A)$ is a weighted coverage function, it inherits the classical properties of such functions.

Proposition 2 (Monotone Submodularity)**.** *The function* $F : 2^{\mathcal{M}} \to \mathbb{R}_{\geq 0}$ *defined in* (4) *is:*

Algorithm 1 GREEDYPORTFOLIO: Greedy Deployment Portfolio Selection

Require: Model archive $\mathcal{M}$, deployment profiles $\mathcal{R} = \{r_1, \ldots, r_M\}$ with weights $\{w_m\}$, budget K
Ensure: Portfolio A with $|A| \leq K$

$A \leftarrow \emptyset$
Precompute $e_r \leftarrow \{x \in \mathcal{M} : x \models r\}$ for all $r \in \mathcal{R}$ {Feasibility sets}
for $k = 1, \ldots, K$ **do**
  for each $x \in \mathcal{M} \setminus A$ **do**
    $\Delta(x) \leftarrow F(A \cup \{x\}) - F(A) = \sum_{m: A \cap e_{r_m} = \emptyset} w_m \cdot \mathbf{1}[x \in e_{r_m}]$
  end for
  $x^* \leftarrow \arg\max_{x \in \mathcal{M} \setminus A} \Delta(x)$
  $A \leftarrow A \cup \{x^*\}$
end for
return A

Algorithm 2 LAZYGREEDYPORTFOLIO: Lazy Evaluation Variant

Require: Model archive $\mathcal{M}$, profiles $\mathcal{R}$, weights $\{w_m\}$, budget K
Ensure: Portfolio A with $|A| \leq K$

$A \leftarrow \emptyset$; Initialize max-heap H with $(\Delta_{\text{init}}(x), x, 0)$ for all $x \in \mathcal{M}$
for $k = 1, \ldots, K$ **do**
  repeat
    Pop $(\bar{\Delta}, x, t)$ from H
    Recompute $\Delta(x) \leftarrow F(A \cup \{x\}) - F(A)$
    if $t < k - 1$ **then**
      Push $(\Delta(x), x, k - 1)$ back to H
    end if
  until $t = k - 1$ {Marginal gain is current}
  $A \leftarrow A \cup \{x\}$
end for
return A

1. ***Monotone**: $F(A) \leq F(B)$ whenever $A \subseteq B$.*
2. ***Submodular**: $F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B)$ whenever $A \subseteq B$ and $x \notin B$.*

By the classical result of Nemhauser, Wolsey, and Fisher [23], the greedy algorithm that iteratively adds the model with the largest marginal coverage gain achieves a $(1 - 1/e)$-approximation.

Theorem 3 (Approximation Guarantee)**.** *Let A^{greedy} be the output of Algorithm 1 and A^* be an optimal solution to* (4)*. Then*

$$F(A^{greedy}) \;\geq\; \left(1 - \frac{1}{e}\right) \cdot F(A^*). \tag{6}$$

4.2 Efficient Implementation via Lazy Evaluation

The naive greedy implementation requires $O(KNM)$ marginal-gain bookkeeping in the worst case. We also implement a lazy-evaluation variant (CELF), which can reduce recomputation on larger or denser instances because marginal gains only decrease under submodularity. On the present HW-NAS-Bench archive and small-K regime, however, heap-maintenance overhead dominates: plain Greedy is faster than Lazy Greedy on the selected family, and lazy evaluation only becomes beneficial once the number of profiles is sufficiently large (Appendix H). We therefore view lazy evaluation as a scaling-oriented implementation option rather than a universally faster default.

4.3 Handling the Distributional Setting

For distributional profiles we run Algorithm 1 on an empirical average over n i.i.d. sampled profiles; generalization is analyzed in the next Section.

5 Theoretical Analysis

Generalization Bound for Profile Coverage. Our main theoretical result shows that the empirical coverage objective concentrates uniformly over all possible portfolios, allowing the sample-then-optimize strategy to generalize.

Theorem 4 (Uniform Convergence of Portfolio Coverage)**.** *Let $\mathcal{M}$ be a model archive of size N with $1 \le K \le N$, let P be a distribution over deployment profiles, and let $r_1, \dots, r_n \sim P$ be i.i.d. samples. For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the draw of the samples:*

$$\sup_{A \subseteq \mathcal{M},\, |A| \le K} \left| F(A) - \hat{F}(A) \right| \;\le\; \sqrt{\frac{K \ln(eN/K) + \ln(2/\delta)}{2n}}. \tag{7}$$

The proof and the corresponding sample-complexity corollary are deferred to Appendix A. The resulting rate scales as $O((K \ln(eN/K) + \ln(1/\delta))/\epsilon^2)$, which is logarithmic in N for the portfolio sizes studied here.

Why bundling can matter. Bundling can matter sharply: Appendix A gives a separation showing that optimizing any single marginal attribute can be arbitrarily suboptimal for bundled profile coverage. Intuitively, marginally strong models may cluster on the same trade-off niche, whereas bundled coverage rewards complementary models that satisfy different profile bundles.

6 Experiments

We evaluate Deployment Portfolio Selection on HW-NAS-Bench in the regime most relevant to our formulation: small portfolios under bundled heterogeneous-device constraints. Our empirical study has two complementary parts. First, we test robustness across a generator sweep of 18 profile families spanning three device pairs, three threshold schedules, and energy on/off. Second, we use one representative EdgeGPU–Raspi4 family for a more detailed case study of coverage curves, hard-profile behavior, non-uniform demand, generalization to held-out profiles, noise robustness, runtime/storage scaling, and reduced-instance exact optimality gaps. This organization separates breadth from interpretability: the family sweep establishes robustness, while the selected family provides readable structural diagnostics.

6.1 Experimental Setup

Offline archive and profile families. We use HW-NAS-Bench [15] on the NAS-Bench-201 search space [11] as a fully offline archive of $N = 15{,}625$ models, combined with CIFAR-10 accuracy (also CIFAR-100 and ImageNet16-120 in Appendix I); portfolio selection reduces to combinatorial subset selection over precomputed attributes. For the robustness sweep we evaluate 18 profile families formed by the Cartesian product of three device pairs (EdgeTPU+FPGA, EdgeGPU+Raspi4, EdgeGPU+Eyeriss), three threshold schedules (Moderate/Moderate, High/Moderate, Very high/Strict), and an energy on/off toggle, each over five profile seeds with $n_{\text{train}} = n_{\text{test}} = 100$. Each profile requires a single model to satisfy an accuracy floor and two device-latency ceilings, plus an energy ceiling in energy-on families (the archive supplies no peak-memory measurements). Excluding $4{,}699$ architectures with missing EdgeTPU latency measurements leaves $10{,}926$ candidates for EdgeTPU families. For detailed visualization we additionally study the representative family `moderate__moderate__edgegpu__raspi4__energy_off` at $K \in \{1, \dots, 5\}$. Appendix B gives the full generator, thresholds, train/test counts, and case-study rationale. Code is available at `https://github.com/jicheng9617/F4MwCons`.

Baselines and metrics. The sweep compares Greedy against FewForMany (composite reduction $\phi_r(x) = \max_{j \in S_r} \text{violation}_{r,j}(x)$), Pareto-$K$ (Pareto-front restriction plus representative selection), and Top-K Accuracy; the selected-family case study additionally reports Top-K Composite, Pareto-Diverse, and Random-K. Let $\mathcal{R}_{\text{cov}} = \{r \in \mathcal{R} : \exists x \in \mathcal{M},\, x \models r\}$ denote coverable profiles. We report the coverable ratio $|\mathcal{R}_{\text{cov}}|/|\mathcal{R}|$, raw coverage on all test profiles, conditional coverage on $\mathcal{R}_{\text{cov}}$, and hardest-bucket coverage on coverable test profiles with feasibility fraction $\le 0.5\%$. Appendix D.1 gives exact baseline implementations, normalization choices, and tie-breaking. Unless noted, experiments use uniform profile weights; Section 6.4 evaluates non-uniform demand.

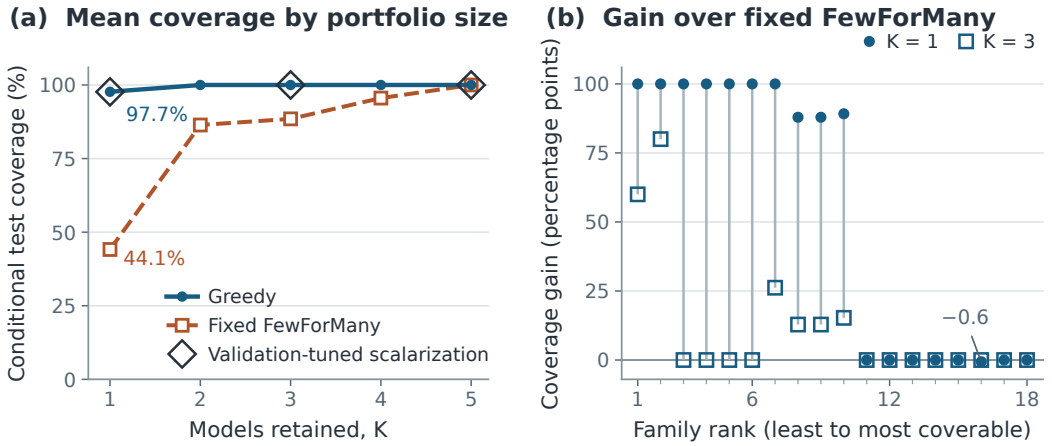


Figure 2: (a) Diamonds mark scalarization with weights, normalization and aggregation chosen on 100 extra validation profiles. (b) Greedy minus fixed FewForMany gaps across the 18 profile families. Positive values favor Greedy. Families are ordered by coverable ratio. The small-K advantage is largest in lower-coverability families and disappears by $K = 5$.

Table 1: Cross-family summary of Greedy versus fixed FewForMany. Positive gaps favor Greedy. The coverability split uses the median family-level coverable ratio (0.623).

Setting	Greedy wins / 18	Mean gap
Conditional coverage @ $K = 1$	10	0.536
Conditional coverage @ $K = 3$	6	0.115
Conditional coverage @ $K = 5$	0	0.000
Hardest-bucket coverage @ $K = 3$	6	0.162
Below-median coverable ratio, cond.@ $K = 1$	–	0.973
Above-median coverable ratio, cond.@ $K = 1$	–	0.098
Below-median coverable ratio, cond.@ $K = 3$	–	0.213
Above-median coverable ratio, cond.@ $K = 3$	–	0.017

6.2 Robustness across profile families

Figure 2 and Table 1 address the main robustness question: does the small-portfolio advantage of direct bundled coverage persist beyond a single selected family? The answer is yes, but in a specific regime. Across the 18 families, Greedy beats fixed FewForMany on conditional coverage in 10/18 families at $K = 1$ and 6/18 families at $K = 3$, with mean gaps of 0.536 and 0.115, respectively. By $K = 5$, the mean gap shrinks to zero, and the methods tie in all 18 families through $K = 10$. The same pattern appears on the hardest bucket at $K = 3$, where Greedy wins in 6/18 families with a mean gap of 0.162.

The advantage is concentrated in lower-coverability families. Splitting the 18 families at the median coverable ratio (0.623), the mean Greedy–FewForMany conditional gap at $K = 1$ is 0.973 below the median and 0.098 above it. At $K = 3$, the corresponding gaps are 0.213 and 0.017. This reinforces the structural interpretation of our method: direct bundled coverage matters most when profile feasibility is sparse and complementary coverage is genuinely required.

The family used for the detailed case study has a coverable ratio of 0.622, close to the sweep median of 0.623. Its Greedy–FewForMany conditional gaps are 0.879 at $K = 1$ and 0.128 at $K = 3$, larger than the corresponding sweep-wide averages of 0.536 and 0.115.

6.3 Representative selected-family case study

Figure 3 shows the same pattern at finer granularity: Greedy leads at $K = 1$ (conditional 0.879 *vs.* 0.000 for FewForMany), the gap narrows at $K = 3$, and the methods are effectively tied by $K = 5$; Pareto-K and Top-K Accuracy remain clearly weaker, confirming that high accuracy alone is a poor proxy for bundled feasibility.

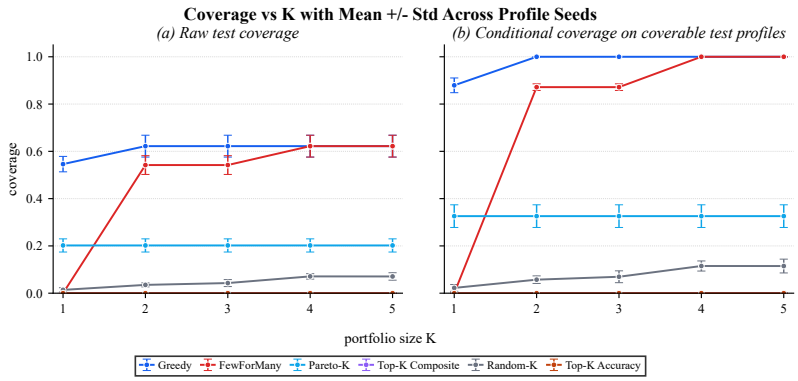


Figure 3: Test raw coverage (left) and conditional coverage on coverable test profiles (right) as functions of portfolio size K on the representative selected HW-NAS-Bench family `moderate__moderate__edgegpu__raspi4__energy_off`. Curves show mean $\pm$ standard deviation across five profile seeds. The same pattern seen in the family sweep reappears at finer resolution: Greedy is strongest in the small-budget regime, while FewForMany becomes competitive as K grows.

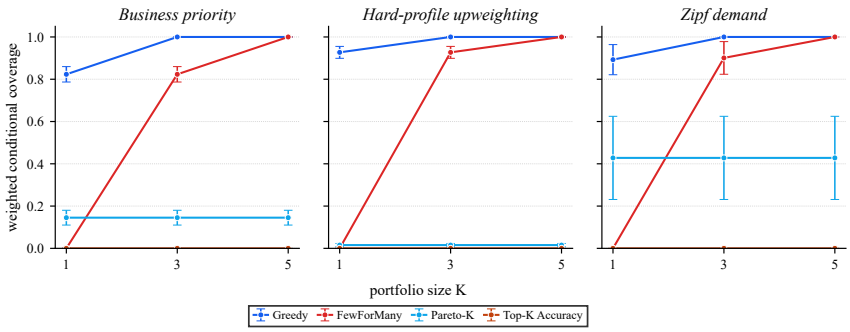


Figure 4: Weighted conditional coverage versus K on the representative selected family under three non-uniform demand models: business priority, hard-profile upweighting, and Zipf demand. Greedy dominates in the small-K regime across all three schemes, while the gap largely collapses by $K = 5$.

Baseline audits and Pareto diagnostics. Per-constraint pass-rate audits (Appendix D.2) show the baseline failures are structural: Top-K Accuracy and Pareto-Diverse have nonzero single-constraint pass rates but zero bundled pass rate, and Top-K Composite has identical conditional coverage across K because later-ranked models add no new bundled coverage. Normalization sensitivity (Appendix D.3) shows the Greedy-vs-FewForMany singleton gap varies with the violation normalization, and the $K = 3$ gap stays positive across threshold-relative, min-max, and robust-IQR scalings. A Pareto-front-restriction diagnostic further separates front restriction from post-hoc selection: at $K = 3$, Greedy and Pareto-front-restricted Greedy differ by 0.000 conditional-coverage points on both the selected family and the 18-family mean, while Pareto-front-restricted Greedy outperforms Pareto-K by 0.674 and 0.836 respectively—so the main Pareto baseline loss comes from the representative-selection heuristic rather than from Pareto restriction itself.

Takeaway. Because part of the profile distribution is infeasible for the archive, high-K raw-coverage gaps are driven by archive infeasibility rather than portfolio construction (conditional-coverage panel of Figure 3). Direct bundled-coverage optimization is most useful in the operational regime with very small portfolios and sparse profile feasibility, and a normalization-robust advantage over fixed scalarization appears at $K = 3$ and on the hardest feasibility bucket. At $K = 1$, Greedy can still outperform scalarized reductions substantially on some families, but the singleton comparison is more sensitive to the exact composite normalization.

Table 2: 18-family aggregate under non-uniform demand. Entries report family-mean weighted conditional coverage (mean $\pm$ std across families); the final column reports how many families Greedy beats FewForMany on weighted conditional coverage.

Scheme	K	Greedy	FewForMany	Pareto-K	Greedy wins / 18
Business priority	1	0.963 ± 0.069	0.420 ± 0.475	0.053 ± 0.078	11
Business priority	3	1.000 ± 0.000	0.782 ± 0.352	0.057 ± 0.077	7
Business priority	5	1.000 ± 0.000	1.000 ± 0.000	0.150 ± 0.215	0
Hard-profile upweighting	1	0.964 ± 0.080	0.332 ± 0.455	0.011 ± 0.018	12
Hard-profile upweighting	3	1.000 ± 0.000	0.765 ± 0.388	0.012 ± 0.018	7
Hard-profile upweighting	5	1.000 ± 0.000	0.889 ± 0.314	0.063 ± 0.115	2
Zipf demand	1	0.979 ± 0.040	0.440 ± 0.472	0.172 ± 0.209	12
Zipf demand	3	1.000 ± 0.000	0.890 ± 0.200	0.194 ± 0.206	7
Zipf demand	5	1.000 ± 0.000	0.989 ± 0.046	0.286 ± 0.293	1

6.4 Weighted Coverage under Non-uniform Profile Importance

Equation (4) allows arbitrary profile weights, but the preceding experiments used uniform-profile evaluation. We therefore evaluate three non-uniform demand models: business-priority weights that upweight selected deployment tiers, hard-profile upweighting that emphasizes low-feasibility profiles, and a Zipf-demand model that concentrates mass on a small set of frequent requests. For each setting we re-optimize each method under the weighted training objective and evaluate weighted raw coverage, weighted conditional coverage, unweighted conditional coverage, and unweighted hardest-bucket coverage on held-out profiles.

Figure 4 and Table 2 show that Greedy's small-K advantage persists under all three schemes. Across the 18 families, Greedy beats FewForMany on weighted conditional coverage in 11–12/18 families at $K = 1$ and 7/18 at $K = 3$ for every weighting scheme; by $K = 5$ the gap largely collapses, although hard-profile upweighting still leaves a nontrivial mean gap (0.111). Pareto-K remains substantially weaker throughout, and Top-K Accuracy stays at zero. On the selected family, Greedy's weighted conditional coverage at $K = 3$ is 1.000 across the three schemes (vs. 0.823–0.927 for FewForMany), while its weighted raw coverage under hard-profile upweighting is only 0.047 because most emphasized profiles are infeasible—i.e., the remaining gap is archive infeasibility, not portfolio construction. Family-mean unweighted conditional coverage stays at 1.000 for $K \geq 3$, indicating little trade-off between prioritized demand and broad coverage. Appendix F gives the full per-scheme tables and demand-weight definitions.

7 Conclusion

We formulated deployment portfolio selection over a fixed pre-measured archive and showed that bundled profile coverage reduces to weighted maximum coverage, inheriting classical greedy guarantees. The empirical picture across an 18-family HW-NAS-Bench sweep is consistent: direct bundled optimization is most valuable in the small-portfolio, low-coverability regime, with gains concentrated on the hardest profiles at $K = 1$–3; by $K = 5$, both methods cover all feasible CIFAR-10 test profiles under uniform demand. This conclusion carries over to three non-uniform demand models. We also study moderate remeasurement noise for $K \geq 3$ (with a simple margin-aware variant further improving chance coverage and reselection stability) and compact feasibility encodings that substantially reduce maintained storage without changing the selected portfolio.

Limitations and future work. Our analysis assumes a fixed archive with deterministic attributes and i.i.d. sampled profiles. Extending it to robust or probabilistic coverage under measurement uncertainty and profile shift is the main open direction. Empirically, evidence is strongest in the small-portfolio regime and on a single offline architecture space, and broader claims about heterogeneous deployment will require validation across additional archives, larger search spaces, and real-device remeasurement. A smaller portfolio simplifies maintenance, but aggregate coverage can leave rare requests underserved. Operators can prioritize them through profile weights (Section 6.4).

Acknowledgments and Disclosure of Funding

This research received no third-party funding. The author declares no financial interests related to this work.

References

[1] H. Benmeziane, K. El Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang. Hardware-aware neural architecture search: Survey and taxonomy. In Z.-H. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4322–4329. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/592. URL `https://doi.org/10.24963/ijcai.2021/592`. Survey Track.

[2] H. Benmeziane, S. Niar, H. Ouarnoughi, and K. El Maghraoui. Pareto rank surrogate model for hardware-aware neural architecture search. In *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 267–276. IEEE, 2022.

[3] H. Benmeziane, H. Ouarnoughi, K. El Maghraoui, and S. Niar. Multi-objective hardware-aware neural architecture search with Pareto rank-preserving surrogate models. *ACM Transactions on Architecture and Code Optimization*, 20(2):1–21, 2023.

[4] H. Cai, L. Zhu, and S. Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HylVB3AqYm`.

[5] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han. Once-for-All: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HylxE1HKwS`.

[6] C. Chekuri, T. Inamdar, K. Quanrud, K. Varadarajan, and Z. Zhang. Algorithms for covering multiple submodular constraints and applications. *Journal of Combinatorial Optimization*, 2022.

[7] A.-C. Cheng, J.-D. Dong, C.-H. Hsu, S.-H. Chang, M. Sun, S.-C. Chang, J.-Y. Pan, Y.-T. Chen, W. Wei, and D.-C. Juan. Searching toward Pareto-optimal device-aware neural architectures. In *Proceedings of the International Conference on Computer-Aided Design*, pages 1–7, 2018.

[8] G. Claeskens and N. L. Hjort. *Model Selection and Model Averaging*. Cambridge University Press, 2008.

[9] B. Doerr, C. Doerr, A. Neumann, F. Neumann, and A. Sutton. Optimization of chance-constrained submodular functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1460–1467, 2020. doi: 10.1609/aaai.v34i02.5504.

[10] J.-D. Dong, A.-C. Cheng, D.-C. Juan, W. Wei, and M. Sun. DPP-Net: Device-aware progressive search for Pareto-optimal neural architectures. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 517–531, 2018.

[11] X. Dong and Y. Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HJxyZkBKDr`.

[12] X. Dong, L. Liu, K. Musial, and B. Gabrys. NATS-Bench: Benchmarking NAS algorithms for architecture topology and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3634–3646, 2021.

[13] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[14] A. Krause and D. Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, pages 71–104. Cambridge University Press, 2014.

[15] C. Li, Z. Yu, Y. Fu, Y. Zhang, Y. Zhao, H. You, Q. Yu, Y. Wang, C. Hao, and Y. Lin. HW-NAS-Bench: Hardware-aware neural architecture search benchmark. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=_0kaDkv3dVf`.

[16] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 510–520, 2011.

[17] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han. MCUNet: Tiny deep learning on IoT devices. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 11711–11722, 2020.

[18] J. Lin, W.-M. Chen, H. Cai, C. Gan, and S. Han. Memory-efficient patch-based inference for tiny deep learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=C1mPUP7uKNp`.

[19] X. Lin, Y. Liu, X. Zhang, F. Liu, Z. Wang, and Q. Zhang. Few for Many: Tchebycheff set scalarization for many-objective optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=O4N9kWwV6R`.

[20] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf. NSGA-Net: Neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 419–427, 2019.

[21] Z. Lu, K. Deb, and V. N. Boddeti. MUXConv: Information multiplexing in convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12044–12053, 2020.

[22] G. Malkomes, B. Cheng, E. H. Lee, and M. McCourt. Beyond the Pareto efficient frontier: Constraint active search for multiobjective experimental design. In *International Conference on Machine Learning*, pages 7423–7434. PMLR, 2021.

[23] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978.

[24] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

[25] M. Staib, B. Wilder, and S. Jegelka. Distributionally robust submodular maximization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 506–516. PMLR, 2019.

[26] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. MnasNet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[27] S. Teerapittayanon, B. McDanel, and H. T. Kung. BranchyNet: Fast inference via early exiting from deep neural networks. In *International Conference on Pattern Recognition*, pages 2464–2469, 2016. doi: 10.1109/ICPR.2016.7900006.

[28] J. Van Delm, M. Vandersteegen, A. Burrello, G. M. Sarda, F. Conti, D. J. Pagliari, L. Benini, and M. Verhelst. HTVM: Efficient neural network deployment on heterogeneous TinyML platforms. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.

[29] K. Wei, R. Iyer, and J. Bilmes. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963. PMLR, 2015.

[30] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

This appendix is organized as follows.

Appendix A collects additional theoretical details.

Appendix B summarizes the experimental setup and the rationale for the selected case-study family.

Appendix C reports additional empirical results, including cross-family tables, hardest-bucket analyses, Pareto-front diagnostics, structural visualizations, and reduced-instance exact-gap experiments.

Appendix D provides baseline, sensitivity, generalization, and selected-family runtime diagnostics.

Appendix E presents additional structural diagnostics for the selected family and the full generator sweep.

Appendix F reports non-uniform-demand weighted-coverage results.

Appendix G reports attribute-noise robustness results.

Appendix H reports scaling and maintained-memory results.

Appendix I studies larger portfolios, validation tuning, alternative tasks, shifted requests and exact scalarization.

A Additional Theoretical and Structural Details

A.1 Hypergraph Interpretation and relation to composite reduction

The deployment coverage problem admits a natural *hypergraph* interpretation that clarifies its combinatorial structure.

Definition 4 (Deployment Hypergraph)**.** *The **deployment hypergraph** $\mathcal{H} = (\mathcal{M}, \mathcal{E})$ has vertex set $\mathcal{M}$ (the model archive) and hyperedge set $\mathcal{E} = \{e_r : r \in \mathcal{R}\}$, where*

$$e_r = \{x \in \mathcal{M} : x \models r\} \tag{8}$$

is the set of all models satisfying profile r.

Under this interpretation, a portfolio A covers profile r if and only if $A \cap e_r \neq \emptyset$. The objective (4) becomes a weighted hyperedge coverage problem: select K vertices to hit as many (weighted) hyperedges as possible.

This hypergraph structure encodes the key distinction from standard multi-objective selection. In a standard Pareto-front approach, one would consider model dominance rather than the profile hyperedges. The *hyper*edges in our formulation capture the bundled nature of profiles: each hyperedge e_r is the *intersection* of multiple per-attribute feasibility sets, and this intersection can behave very differently from the individual sets.

Remark 1. *One might ask whether the hypergraph formulation is necessary: if $|\mathcal{R}|$ is small, one could define a composite objective $\phi_{x,r} = \max_{j \in S_r} \text{violation}_{r,j}(x)$ and reduce to a standard "few-for-many" problem over scalar objectives $\{\phi_r\}$. We acknowledge this reduction is valid. However, in realistic deployment settings, the number of profiles M can be large (hundreds to thousands, arising from combinatorial combinations of devices, latency tiers, and memory classes), and the* distributional *version* (5) *is more natural. Our formulation and theory directly handle this regime.*

A.2 Proof of Theorem 1

Proof. We reduce from Maximum k-Coverage, which is known to be NP-hard [13]. An instance of Maximum k-Coverage consists of a universe $U = \{u_1, \ldots, u_M\}$, a collection of sets $\mathcal{S} = \{S_1, \ldots, S_N\}$ with $S_i \subseteq U$, and a budget K. The goal is to select K sets from $\mathcal{S}$ to maximize the number of covered elements.

Given such an instance, we construct a Deployment Portfolio Selection instance as follows. Define N models $\mathcal{M} = \{x_1, \ldots, x_N\}$ with a single binary attribute per universe element: $a_m(x_i) = \mathbf{1}[u_m \in S_i]$ for $m = 1, \ldots, M$. Define M deployment profiles, one per universe element: profile r_m has $S_{r_m} = \{m\}$, $\tau_{r_m,m} = 1$, $\sigma_{r_m,m} = +1$. Then $x_i \models r_m$ if and only if $u_m \in S_i$. The coverage objective (4) with uniform weights becomes exactly the number of covered universe elements. Hence, any polynomial-time algorithm for Deployment Portfolio Selection with approximation ratio better than $1 - 1/e + \epsilon$ would violate the hardness of Maximum k-Coverage under standard assumptions. □

A.3 Proof of Theorem 4

Proof. The proof proceeds in two steps.

Step 1: Counting portfolios. The number of distinct portfolios of size at most K is $\sum_{k=0}^{K} \binom{N}{k} \leq \left(\frac{eN}{K}\right)^K$.

Step 2: Uniform convergence via union bound and Hoeffding. Fix a portfolio A with $|A| \leq K$. Since $\text{cov}(A, r_i) \in \{0, 1\}$ and the r_i are i.i.d., by Hoeffding's inequality:

$$\mathbb{P}\big[|F(A) - \hat{F}(A)| > \epsilon\big] \;\leq\; 2\exp(-2n\epsilon^2). \tag{9}$$

Taking a union bound over all $\leq (eN/K)^K$ portfolios:

$$\mathbb{P}\Big[\sup_{|A|\leq K} |F(A) - \hat{F}(A)| > \epsilon\Big] \;\leq\; 2\Big(\frac{eN}{K}\Big)^K \exp(-2n\epsilon^2). \tag{10}$$

Setting the right-hand side equal to δ and solving for ϵ yields the bound (7). □

A.4 Sample-complexity corollary

Corollary 5 (Sample Complexity)**.** *To ensure that the empirical greedy portfolio $\hat{A}_n$ satisfies $F(\hat{A}_n) \geq (1 - 1/e) \cdot F(A^*) - \epsilon$ with probability at least $1 - \delta$, it suffices to use*

$$n \;=\; O\Big(\frac{K\ln(eN/K) + \ln(1/\delta)}{\epsilon^2}\Big) \tag{11}$$

sampled profiles.

Remark 2 (Interpretation)**.** *The sample complexity scales as $K\ln(eN/K)$, which is the "effective dimension" of the portfolio selection problem. For typical values ($N = 15{,}625$, $K = 5$), this is approximately $5 \times \ln(3125e) \approx 45$. At $\delta = 0.05$, the bound is 0.495 at $n = 100$ and 0.221 at $n = 500$. This is important in practice: exhaustively enumerating all possible deployment profiles (combinatorial in the number of devices $\times$ constraint levels) is infeasible, but sampling a moderate number is easy.*

Proof. By Theorem 4, with probability $\geq 1 - \delta$, $|\hat{F}(A) - F(A)| \leq \epsilon/2$ for all $|A| \leq K$. The greedy algorithm on $\hat{F}$ returns $\hat{A}_n$ with $\hat{F}(\hat{A}_n) \geq (1 - 1/e) \cdot \max_{|A|\leq K} \hat{F}(A)$. Combining:

$$\begin{aligned} F(\hat{A}_n) &\geq \hat{F}(\hat{A}_n) - \epsilon/2 \geq (1 - 1/e)\hat{F}(A^*) - \epsilon/2 \\ &\geq (1 - 1/e)(F(A^*) - \epsilon/2) - \epsilon/2 \geq (1 - 1/e)F(A^*) - \epsilon. \quad \square \end{aligned} \tag{12}$$

A.5 Arbitrary separation for single-attribute selection

Proposition 6 (Arbitrary separation for single-attribute selection)**.** *For every integer $K \geq 2$, there exists an instance of Deployment Portfolio Selection with budget K such that the portfolio obtained by selecting the K models with the largest value of a single attribute covers only a $1/K$ fraction of the profiles covered by an optimal bundled-coverage portfolio.*

Proof. Let the attributes be $q, b_1, \ldots, b_K$, all with direction $+1$. Create $2K$ models: (i) decoys d_i for $i = 1, \ldots, K$ with $q(d_i) = 2 + i/K$, $b_1(d_i) = 1$, and $b_j(d_i) = 0$ for all $j > 1$; (ii) specialists s_i for $i = 1, \ldots, K$ with $q(s_i) = 1$, $b_i(s_i) = 1$, and $b_j(s_i) = 0$ for all $j \neq i$. Create K equally weighted deployment profiles r_i with active attributes $S_{r_i} = \{q, b_i\}$ and thresholds $q \geq 1$, $b_i \geq 1$.

Then each specialist s_i satisfies exactly profile r_i, whereas every decoy d_i satisfies only r_1. Since $q(d_i) > q(s_j)$ for all i, j, the top-K portfolio by the single attribute q is $\{d_1, \ldots, d_K\}$ and covers only one profile. By contrast, the bundled-optimal portfolio $\{s_1, \ldots, s_K\}$ covers all K profiles. Hence the single-attribute portfolio achieves only a $1/K$ fraction of the optimal bundled coverage.

Thus, the approximation ratio of top-K selection by a single marginal attribute can be as small as $1/K$, and hence can be made arbitrarily close to zero as K grows. □

A.6 Connection to Fairness

The Deployment Portfolio Selection framework has a natural analogy in algorithmic fairness. Consider a setting where:

- $\mathcal{M}$ is a set of candidate classifiers (e.g., from a Rashomon set of near-optimal models);
- A "deployment profile" becomes a *fairness profile*: a subgroup (e.g., defined by intersecting demographic attributes) together with performance thresholds (e.g., minimum recall ≥ 0.80 and maximum false positive rate ≤ 0.10);
- The goal is to select K models such that each subgroup is well-served by at least one model.

This "fairness portfolio" formulation is relevant when an organization deploys different models for different user populations (e.g., region-specific models) and wants to ensure that every demographic subgroup has access to a model meeting fairness requirements.

The algorithmic and theoretical results of this paper apply directly: the fairness coverage objective is monotone submodular, the greedy algorithm provides a $(1 - 1/e)$-approximation, and the generalization bound applies when fairness profiles are drawn from a distribution (e.g., randomly intersecting demographic attributes).

B Additional Experimental Setup

B.1 Offline archive and attributes

The primary experiments use a fully offline archive derived from HW-NAS-Bench [15] combined with CIFAR-10 accuracy from NAS-Bench-201 [11]. The resulting archive contains $N = 15{,}625$ candidate architectures from the NAS-Bench-201 search space. Because all quantities are pre-measured, portfolio selection requires no retraining and reduces to combinatorial subset selection over a fixed archive.

Table 3 summarizes the paper-facing attribute dictionary. Accuracy is always treated as a benefit attribute, while latency and energy are always treated as cost attributes. In every CIFAR-10 profile family, the active attribute set includes CIFAR-10 accuracy and the two latency attributes associated with the selected device pair; energy is included only in energy-on families.

B.2 Profile-family generator and evaluation protocol

All paper-facing profile-family experiments use the same fixed offline archive described in Appendix B.1, with $N = 15{,}625$ candidate models (10,926 for EdgeTPU). Accuracy thresholds are expressed on the $[0, 1]$ scale, latency thresholds in milliseconds (ms), and energy thresholds in millijoules (mJ).

Section 6.2 evaluates 18 profile families formed by the Cartesian product of three device pairs (`EdgeTPU+FPGA`, `EdgeGPU+Raspi4`, `EdgeGPU+Eyeriss`), three threshold schedules (`Moderate/Moderate`, `High/Moderate`, `Very high/Strict`), and an energy on/off toggle. Each profile requires a *single* model to satisfy an accuracy floor together with latency ceilings on two devices; in energy-on families, an additional energy ceiling is included.

Generator. We index a profile family by

$$f = (h_1, h_2, q, e), \tag{13}$$

where (h_1, h_2) is the active device pair, $q \in \{\mathrm{M/M}, \mathrm{H/M}, \mathrm{VH/S}\}$ is the threshold schedule, and $e \in \{\mathrm{on}, \mathrm{off}\}$ is the energy toggle. The active latency attributes and energy-attribute mapping are given in Table 4. All families use the same two-stage threshold generator.

First, a shared difficulty label is drawn once per sampled profile:

$$d \sim \mathrm{Categorical}\big(\mathrm{easy} : 0.34,\ \mathrm{medium} : 0.33,\ \mathrm{hard} : 0.33\big). \tag{14}$$

Conditional on d, the accuracy threshold and all active cost thresholds are sampled *independently* from finite grids:

$$\tau_{\mathrm{acc}} \sim \mathrm{Unif}(A_{q,d}), \tag{15}$$

Table 3: Attribute dictionary for the offline archive used in all paper-facing experiments. Accuracy is on the $[0, 1]$ scale; latency is in milliseconds (ms); energy is in millijoules (mJ). ET+F = EdgeTPU+FPGA, EG+R = EdgeGPU+Raspi4, and EG+E = EdgeGPU+Eyeriss.

Attribute	Benchmark / metric	Unit	Dir.	Active families	Notes
`accuracy`	NAS-Bench-201 / CIFAR-10 test accuracy	$[0, 1]$	↑	All 18	Retrieved via the official NAS-Bench-201 API as `test-accuracy`/100 with `dataset='cifar10'`, `hp='200'`, and `is_random=False`; equivalently, averaged `ori-test` accuracy at 200 epochs.
`edgetpu_latency`	HW-NAS-Bench / EdgeTPU latency	ms	↓	All ET+F	Active only for the EdgeTPU+FPGA device pair.
`fpga_latency`	HW-NAS-Bench / FPGA latency	ms	↓	All ET+F	Active only for the EdgeTPU+FPGA device pair.
`edgegpu_latency`	HW-NAS-Bench / EdgeGPU latency	ms	↓	All EG+R, EG+E	Shared latency attribute across both EdgeGPU-based device pairs.
`raspi4_latency`	HW-NAS-Bench / Raspi4 latency	ms	↓	All EG+R	Active only for the EdgeGPU+Raspi4 device pair.
`eyeriss_latency`	HW-NAS-Bench / Eyeriss latency	ms	↓	All EG+E	Active only for the EdgeGPU+Eyeriss device pair.
`fpga_energy`	HW-NAS-Bench / FPGA energy	mJ	↓	Energy-on ET+F	Included only when the energy toggle is on.
`edgegpu_energy`	HW-NAS-Bench / EdgeGPU energy	mJ	↓	Energy-on EG+R	Included only when the energy toggle is on.
`eyeriss_energy`	HW-NAS-Bench / Eyeriss energy	mJ	↓	Energy-on EG+E	Included only when the energy toggle is on.

$$\tau_j \sim \mathrm{Unif}(C_{j,c(q),d}) \qquad \text{independently for each active cost attribute } j, \tag{16}$$

where the cost-schedule map is

$$c(q) = \begin{cases} \text{moderate}, & q \in \{\mathrm{M/M}, \mathrm{H/M}\}, \\ \text{strict}, & q = \mathrm{VH/S}. \end{cases} \tag{17}$$

Thus, for a family $f = (h_1, h_2, q, e)$, the sampled profile always includes accuracy and the two family-selected latency constraints, and additionally includes the family-selected energy constraint when $e = \text{on}$. Every sampled profile includes all constraints active in its family. The constraint directions are fixed as

$$\sigma^r_{\text{acc}} = +1, \qquad \sigma^r_j = -1 \ \text{ for every latency or energy attribute } j. \tag{18}$$

Appendix I describes how the same percentile levels are used across tasks. The full 18-family generator is specified by the device-pair map in Table 4, the accuracy grids in Table 5, and the cost grids in Table 6.

Table 4: Device-pair map used by the 18 profile families.

Device pair (h_1, h_2)	Active latency attributes	Energy attribute if $e = \text{on}$
`EdgeTPU+FPGA`	`edgetpu_latency`, `fpga_latency`	`fpga_energy`
`EdgeGPU+Raspi4`	`edgegpu_latency`, `raspi4_latency`	`edgegpu_energy`
`EdgeGPU+Eyeriss`	`edgegpu_latency`, `eyeriss_latency`	`eyeriss_energy`

Protocol. For the 18-family robustness sweep, all families use the same seed set and profile counts:

$$u \in \{42, 43, 44, 45, 46\}, \qquad n_{\text{train}} = 100, \qquad n_{\text{test}} = 100. \tag{19}$$

Table 5: Accuracy-floor grids $A_{q,d}$ by threshold schedule q and difficulty label d. Each sampled accuracy threshold is drawn uniformly from the corresponding finite set. Values shown use the full CIFAR-10 archive.

Threshold schedule q	easy	medium	hard
`Moderate/Moderate`	{0.914767, 0.920300, 0.921850}	{0.921850, 0.923650, 0.926100}	{0.926100, 0.929360, 0.931400}
`High/Moderate`	{0.921850, 0.923650, 0.926100}	{0.926100, 0.929360, 0.931400}	{0.931400, 0.932793, 0.934600}
`Very high/Strict`	{0.926100, 0.929360, 0.931400}	{0.931400, 0.932793, 0.934600}	{0.934600, 0.936100, 0.937750}

Table 6: Latency- and energy-threshold grids $C_{j,s,d}$ by attribute j, cost schedule s, and difficulty label d. Each sampled cost threshold is drawn uniformly from the corresponding finite set on eligible architectures.

Attribute j	easy	medium	hard
Moderate cost schedule			
`edgetpu_latency`	{1.04204154, 1.19041090, 1.28812253}	{0.87758741, 0.98782230, 1.12485202}	{0.74884491, 0.82741873, 0.87758741}
`fpga_latency`	{3.72496000, 3.93616000, 4.24016000}	{3.20976000, 3.42096000, 3.83056000}	{2.87696000, 2.98256000, 3.20976000}
`edgegpu_latency`	{5.09916496, 5.48823452, 5.80302429}	{4.39899635, 4.93498707, 5.31205177}	{3.76162815, 4.13753033, 4.39899635}
`raspi4_latency`	{24.83207626, 26.57036799, 30.22856031}	{19.28647704, 23.43833948, 25.68010678}	{12.92314194, 15.72282284, 19.28647704}
`eyeriss_latency`	{5.12018000, 6.10322000, 6.34898000}	{4.87442000, 4.87442000, 5.12018000}	{3.89138000, 4.62866000, 4.87442000}
`fpga_energy`	{26.07472000, 27.55312000, 29.68112000}	{22.46832000, 23.94672000, 26.81392000}	{20.13872000, 20.87792000, 22.46832000}
`edgegpu_energy`	{23.06863257, 24.78686426, 26.23389085}	{19.93150374, 22.13409412, 23.95404318}	{16.98944062, 18.63528260, 19.93150374}
`eyeriss_energy`	{1.00049500, 1.18043291, 1.24707008}	{0.93799955, 0.97338899, 1.03174273}	{0.72267220, 0.93385784, 0.93799955}
Strict cost schedule			
`edgetpu_latency`	{0.87758741, 0.92979886, 0.98782230}	{0.74884491, 0.82741873, 0.87758741}	{0.60618337, 0.69459896, 0.74884491}
`fpga_latency`	{3.20976000, 3.31536000, 3.42096000}	{2.87696000, 2.98256000, 3.20976000}	{2.39056000, 2.57296000, 2.87696000}
`edgegpu_latency`	{4.39899635, 4.69338512, 4.93498707}	{3.76162815, 4.13753033, 4.39899635}	{3.03424549, 3.36882114, 3.76162815}
`raspi4_latency`	{19.28647704, 22.12075081, 23.43833948}	{12.92314194, 15.72282284, 19.28647704}	{8.04503974, 10.56046056, 12.92314194}
`eyeriss_latency`	{4.87442000, 4.87442000, 4.87442000}	{3.89138000, 4.62866000, 4.87442000}	{3.39986000, 3.64562000, 3.89138000}
`fpga_energy`	{22.46832000, 23.20752000, 23.94672000}	{20.13872000, 20.87792000, 22.46832000}	{16.73392000, 18.01072000, 20.13872000}
`edgegpu_energy`	{19.93150374, 21.11917825, 22.13409412}	{16.98944062, 18.63528260, 19.93150374}	{13.30424756, 15.10737525, 16.98944062}
`eyeriss_energy`	{0.93799955, 0.96924728, 0.97338899}	{0.72267220, 0.93385784, 0.93799955}	{0.66017675, 0.69142448, 0.72267220}

For a run indexed by seed u, the training profiles are sampled with RNG seed u, and the test profiles are sampled independently with RNG seed $u + 1$. Adjacent runs reuse streams across roles. The implementation does not enforce tuple-level deduplication within train, within test, or across train/test; duplicates are therefore allowed. All sampled profiles enter the evaluation without rejection, clipping or filtering by coverability. Estimating the coverable fraction of requests in a deployed fleet would require logs from that fleet. Because thresholds are drawn from finite empirical archive quantiles, out-of-range handling is not needed. Thresholds are represented in `float64` during generation; exported `profiles_*.csv` files are formatted to eight decimal places only at write time.

The family sweep uses $K \in \{1, 3, 5\}$ (curves through $K = 10$ in Figure 15a) and the method subset Greedy, FewForMany, Pareto-K, and Top-K Accuracy. For the representative selected family `moderate__moderate__edgegpu__raspi4__energy_off`, we additionally report full coverage curves over $K \in \{1, 2, 3, 4, 5\}$ and the expanded baseline suite. Representative structural figures in the appendix use the seed-42 run. In the representative seed-42 split shown in Figure 6, the coverable test profiles are distributed as 15 in the $\leq 0.5\%$ bucket, 8 in the $0.5\% - 1\%$ bucket, 28 in the $1\% - 5\%$ bucket, and 12 in the $> 5\%$ bucket.

Implementation note. Some `edgetpu_latency` records contain the archive value -1.00000000. These missing measurements are excluded from the empirical quantiles for that attribute. Valid thresholds are used verbatim by the generator; they are not introduced by clipping or any other post-processing.

B.3 Rationale for the selected case-study family

In the moderate, energy-off EdgeGPU–Raspi4 family, an additional model can cover requests that the first selected model misses. We use this family for the structural diagnostics and reduced-instance experiment (Appendix C.6). Figure 5 shows the five candidate families examined in the generator diagnostic.

B.4 Metrics and additional protocol details

We report: (i) *coverable ratio*, i.e., the fraction of sampled profiles that are satisfiable by at least one archive model; (ii) *raw coverage*, i.e., the fraction of all test profiles covered by the selected portfolio; (iii) *conditional coverage*, i.e., coverage restricted to the subset of coverable test profiles; and (iv)

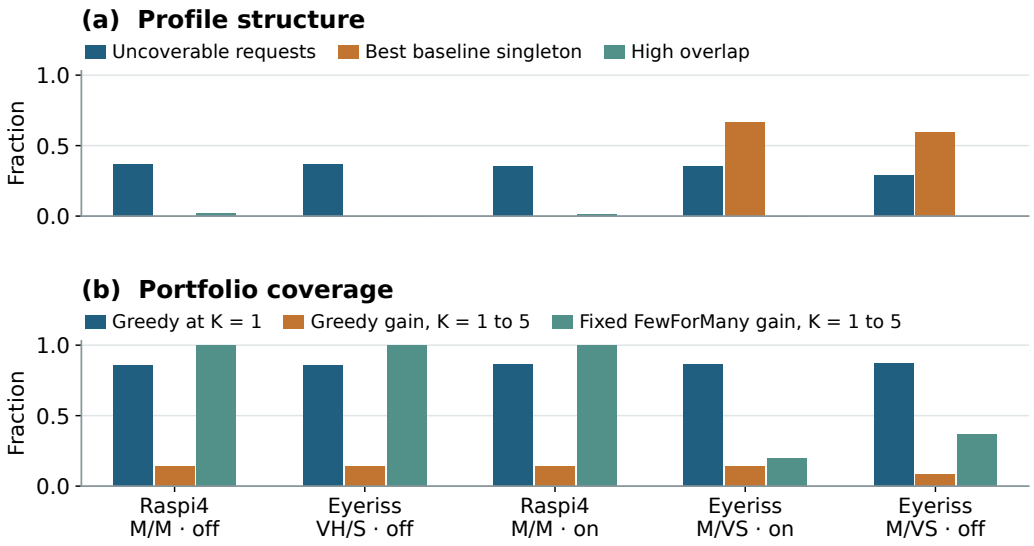


Figure 5: Five generator families on the seed-42 split. All pair EdgeGPU with the device named on the axis. M/M, VH/S and M/VS denote moderate/moderate, very high/strict and moderate/very strict accuracy/cost schedules. On/off indicates the energy constraint. (a) The fractions of all test requests that are uncoverable or feasible for at least 10% of the archive (high overlap), and the better conditional singleton coverage of Top-K Accuracy and fixed FewForMany. (b) Greedy's conditional coverage at $K = 1$ and the gain in conditional coverage for each method from $K = 1$ to $K = 5$.

hardest-bucket coverage, defined on the subset of coverable test profiles with feasibility fraction at most 0.5%. We also visualize representative profile–model coverage heatmaps for structural interpretation.

Appendix D.4 evaluates generalization to held-out profiles under the selected family. There we vary the number of sampled training profiles $n \in \{50, 100, 200, 500\}$, evaluate portfolio sizes $K \in \{3, 5\}$, and report results on 1000 test profiles. Each training-profile count is averaged over five seeds.

All experiments operate on the pre-measured offline archive; no model retraining or hardware remeasurement is needed. The reduced-instance exact-gap experiment uses stratified archive subsampling over accuracy, EdgeGPU latency, and Raspi4 latency. Runtime measurements reported in Appendix D.5 are measured under the same local execution environment for all methods (Intel Core i5-14600KF, 32 GB RAM).

C Additional Experimental Results

C.1 Cross-family robustness details

Table 7 gives the full family-wise view behind Figure 2. On CIFAR-10, all six EdgeTPU–FPGA families tie at $K = 3, 5$.

Appendix I compares the same family grid across tasks.

C.2 Expanded selected-family results

Table 8 expands the selected-family case study from the main text. The full comparison confirms the same qualitative pattern as the family sweep: Top-K Accuracy fails completely under bundled constraints, Pareto-based baselines remain substantially weaker than the leading profile-aware methods, and the most competitive comparison is between Greedy and FewForMany. Greedy is strongest at small portfolio sizes, while FewForMany becomes tied at $K = 5$.

Table 7: Conditional test coverage by profile family, averaged over five seeds. Families are listed in the order used in Figure 2. Cov. is the coverable ratio. G, F and P denote Greedy, fixed FewForMany and Pareto-K, and Δ is G minus F. Hardest denotes coverable profiles feasible for at most 0.5% of the archive.

Profile family				$K = 1$			$K = 3$			Hardest, $K = 3$		
Sched.	Devices	Energy	Cov.	G	F	Δ	G	F	Δ	G	F	P
VH/S	EdgeGPU + Raspi4	off	0.202	1.000	0.000	1.000	1.000	0.400	0.600	1.000	0.400	0.000
VH/S	EdgeGPU + Raspi4	on	0.226	1.000	0.000	1.000	1.000	0.200	0.800	1.000	0.200	0.000
VH/S	EdgeTPU + FPGA	on	0.400	1.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	0.000
VH/S	EdgeTPU + FPGA	off	0.456	1.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	0.000
VH/S	EdgeGPU + Eyeriss	on	0.528	1.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	0.000
H/M	EdgeGPU + Raspi4	off	0.536	1.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	0.000
H/M	EdgeGPU + Raspi4	on	0.556	1.000	0.000	1.000	1.000	0.738	0.262	1.000	0.600	0.000
M/M	EdgeGPU + Raspi4	off	0.622	0.879	0.000	0.879	1.000	0.872	0.128	1.000	0.584	0.000
VH/S	EdgeGPU + Eyeriss	off	0.622	0.879	0.000	0.879	1.000	0.872	0.128	1.000	0.829	0.000
M/M	EdgeGPU + Raspi4	on	0.624	0.892	0.000	0.892	1.000	0.847	0.153	1.000	0.464	0.000
H/M	EdgeTPU + FPGA	on	0.750	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000
M/M	EdgeTPU + FPGA	on	0.804	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000
H/M	EdgeTPU + FPGA	off	0.824	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000
H/M	EdgeGPU + Eyeriss	on	0.876	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000
M/M	EdgeTPU + FPGA	off	0.880	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000
H/M	EdgeGPU + Eyeriss	off	0.958	0.935	0.941	-0.006	1.000	1.000	0.000	1.000	1.000	0.000
M/M	EdgeGPU + Eyeriss	off	1.000	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000
M/M	EdgeGPU + Eyeriss	on	1.000	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.034

M/M = moderate/moderate, H/M = high/moderate, VH/S = very high/strict (accuracy/cost schedules).

Table 8: Expanded raw and conditional test coverage results for the representative selected HW-NAS-Bench family at $K \in \{1, 3, 5\}$. Numbers are mean $\pm$ standard deviation over five profile seeds.

Method	Raw@1	Cond.@1	Raw@3	Cond.@3	Raw@5	Cond.@5
Random-K	0.014 ± 0.009	0.022 ± 0.014	0.043 ± 0.015	0.070 ± 0.025	0.071 ± 0.016	0.115 ± 0.029
Top-K Accuracy	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Top-K Composite	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Pareto-K	0.202 ± 0.028	0.326 ± 0.048	0.202 ± 0.028	0.326 ± 0.048	0.202 ± 0.028	0.326 ± 0.048
Pareto-Diverse	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
FewForMany	0.000 ± 0.000	0.000 ± 0.000	0.542 ± 0.040	0.872 ± 0.014	$\mathbf{0.622 \pm 0.046}$	$\mathbf{1.000 \pm 0.000}$
Greedy	$\mathbf{0.546 \pm 0.033}$	$\mathbf{0.879 \pm 0.031}$	$\mathbf{0.622 \pm 0.046}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{0.622 \pm 0.046}$	$\mathbf{1.000 \pm 0.000}$

C.3 Pareto-front-restricted Greedy diagnostic

Table 9: Conditional coverage for the selected family and the 18-family sweep. PF-restricted Greedy runs the coverage selector on the Pareto front. Comparing it with Pareto-K isolates the effect of representative selection within that front.

Method	Cond.@1	Cond.@3	Cond.@5	Hardest@3
Selected family				
Greedy	0.879 ± 0.031	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
PF-restricted Greedy	0.879 ± 0.031	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Pareto-K	0.326 ± 0.048	0.326 ± 0.048	0.326 ± 0.048	0.000 ± 0.000
Pareto-Diverse	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
18-family sweep				
Greedy	0.977 ± 0.044	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
PF-restricted Greedy	0.977 ± 0.044	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Pareto-K	0.147 ± 0.184	0.164 ± 0.183	0.259 ± 0.284	0.002 ± 0.008
Pareto-Diverse	0.000 ± 0.000	0.003 ± 0.009	0.177 ± 0.369	0.000 ± 0.000

At $K = 3$, Greedy and Pareto-front-restricted Greedy differ by 0.000 conditional-coverage points on both the selected family and the 18-family mean. By contrast, Pareto-front-restricted Greedy outperforms Pareto-K by 0.674 on the selected family and by 0.836 on the 18-family mean. This

indicates that the main Pareto baseline loss comes from the post-hoc representative/diversity heuristic rather than from restricting candidates to the Pareto front itself.

C.4 Hardest-bucket analysis

Table 10: Coverage on the hardest feasibility bucket at $K = 3$ on the representative selected family, aggregated over five profile seeds. The hardest bucket consists of coverable test profiles whose feasibility fraction is at most 0.5%.

Method	Hardest-bucket coverage @ $K = 3$
Greedy	**1.000 ± 0.000**
FewForMany	0.584 ± 0.115
Pareto-K	0.000 ± 0.000

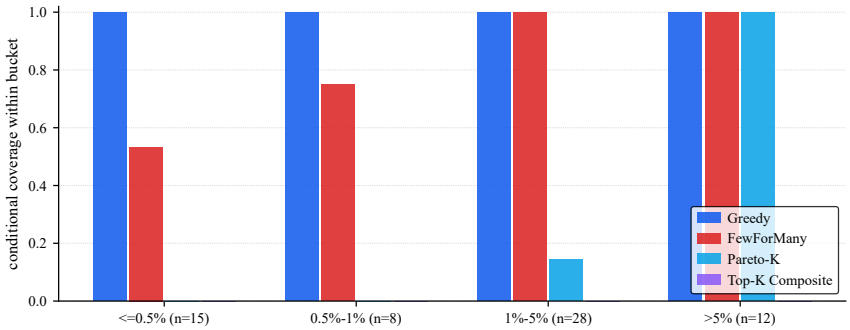


Figure 6: Test coverage at $K = 3$ stratified by profile feasibility on the representative selected family. Bucket labels include the number of coverable test profiles in a representative split. Greedy's advantage is concentrated on the lowest-feasibility profiles, while methods become closer on easier buckets.

The hardest-bucket size is reasonably stable across seeds, with mean 14.4 profiles, minimum 10, and maximum 17. This supports the interpretation of Figure 6: Greedy's advantage on difficult bundled profiles is not an artifact of a single representative split, but persists under repeated resampling.

Figure 6 stratifies the coverable test profiles by feasibility. The strongest separation appears on the hardest bucket, where at most 0.5% of archive models satisfy the profile. On this bucket, Greedy attains perfect coverage in the representative split, while FewForMany covers fewer profiles and Pareto-K degrades sharply. Across five seeds, Greedy achieves 1.000 ± 0.000 hardest-bucket coverage at $K = 3$, compared with 0.584 ± 0.115 for FewForMany and 0.000 ± 0.000 for Pareto-K. These results show that the main benefit of direct bundled-coverage optimization is not a uniform lift on easy profiles, but a concentrated advantage on exactly the sparse, difficult profiles where complementary coverage matters most.

C.5 Selected-family structural visualizations

Figure 7 visualizes the complementary coverage structure behind the hardest-bucket gains. Greedy selects models whose coverage patterns are more complementary: different portfolio members are responsible for different regions of the profile space, especially among lower-feasibility profiles. In contrast, Pareto-K allocates more of the portfolio to overlapping easy-profile coverage, which reduces its effectiveness on the bundled hard-profile regime.

C.6 Exact optimality gap details

To quantify how close Greedy is to the exact maximum-coverage optimum, we solve reduced deployment instances exactly with a binary ILP for weighted maximum coverage. The reduced-instance experiment uses the representative selected family `moderate__moderate__edgegpu__raspi4__energy_off`. We consider reduced archive

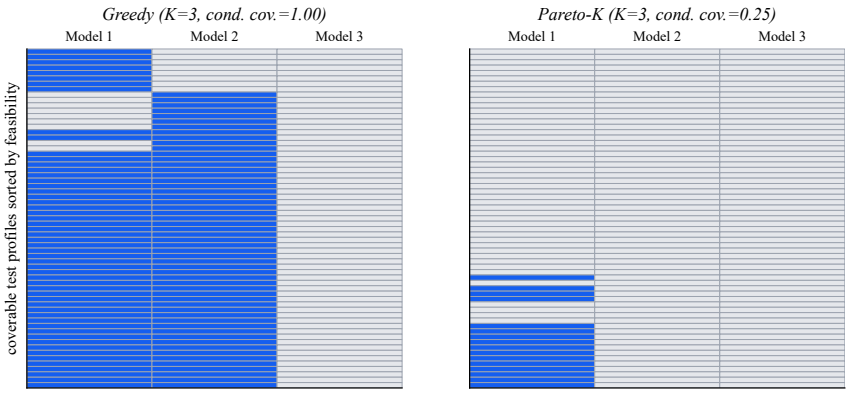


Figure 7: Representative test-split coverage heatmaps at $K = 3$ for Greedy and Pareto-K on the selected family. Rows are deployment profiles sorted by feasibility; columns are the selected models. Greedy allocates portfolio capacity more complementarily, while Pareto-K exhibits more redundant coverage of easier profiles and leaves harder profiles uncovered.

sizes $|\mathcal{M}'| \in \{100, 250, 500\}$, reduced profile counts $|\mathcal{R}'| \in \{100, 250\}$, portfolio sizes $K \in \{1, 3, 5\}$, and seeds $42, \ldots, 51$. Archive subsampling is stratified by joint quantile bins over accuracy, EdgeGPU latency, and Raspi4 latency. This yields 180 reduced instances in total.

For a reduced archive $\mathcal{M}'$ and reduced training profile set $\mathcal{R}'$, let

$$e_r = \{x \in \mathcal{M}' : x \models r\}$$

denote the reduced feasibility set of profile r. We solve the exact reduced problem as the following binary ILP:

$$\max_{y,z} \quad \sum_{r \in \mathcal{R}'} w_r z_r \tag{20}$$

$$\text{s.t.} \quad \sum_{x \in \mathcal{M}'} y_x \leq K, \tag{21}$$

$$z_r \leq \sum_{x \in e_r} y_x, \qquad \forall r \in \mathcal{R}', \tag{22}$$

$$y_x \in \{0, 1\}, \qquad \forall x \in \mathcal{M}', \tag{23}$$

$$z_r \in \{0, 1\}, \qquad \forall r \in \mathcal{R}'. \tag{24}$$

Here y_x indicates whether model x is selected and z_r indicates whether profile r is covered. Unless otherwise stated, the reduced-instance exact-gap experiment uses uniform profile weights on the reduced training set, i.e., $w_r \equiv 1$. Because the objective maximizes covered profile weight, constraint (22) is sufficient to force $z_r = 1$ exactly when at least one selected model satisfies profile r.

We solve the ILP using HiGHS 1.8.0 through SciPy 1.15.3, with automatic thread selection, a relative optimality gap tolerance of 10^{-4} and no per-instance time limit. We report instances for which the solver returned an optimal solution.

Table 11: Exact-gap summary by portfolio size on reduced instances from the selected family. Numbers are reduced-train coverage normalized by the exact ILP optimum.

K	Greedy / ILP	Greedy worst	FewForMany / ILP	FewForMany worst
1	1.0000 ± 0.0000	1.0000	0.1086 ± 0.1564	0.0000
3	1.0000 ± 0.0000	1.0000	0.9496 ± 0.0803	0.7279
5	1.0000 ± 0.0000	1.0000	0.9939 ± 0.0234	0.8909

Across all 180 reduced instances, Greedy matches the exact ILP optimum, with overall ratio 1.0000 and worst-case ratio 1.0000. FewForMany is far from optimal at $K = 1$, becomes much closer by

Table 12: Mean wall-clock runtime in milliseconds for the reduced-instance exact-gap experiment.

Method	$K = 1$	$K = 3$	$K = 5$
Greedy	0.11	0.15	0.20
FewForMany	1.05	2.94	4.92
ILP	4.53	3.49	3.62

$K = 3$, and is nearly optimal by $K = 5$. Appendix I compares the exact maximum-coverage and fixed-scalarization objectives on two device pairs.

The exact solver remains inexpensive on these reduced instances, which makes the optimality-gap check easy to verify empirically.

D Baseline, Sensitivity, Generalization, and Runtime Diagnostics

D.1 Baseline protocol

All baselines operate on the run-specific dataset-filtered archive $\mathcal{M}$, with $|\mathcal{M}| = 15{,}625$ (or $10{,}926$ when EdgeTPU is active) candidate models. For the 18-family robustness sweep, each family is evaluated over profile seeds $u \in \{42, 43, 44, 45, 46\}$; for seed u, we sample $|R_{\text{train}}| = 100$ training profiles with RNG seed u and $|R_{\text{test}}| = 100$ test profiles with RNG seed $u+1$. Greedy, FewForMany, Pareto-K, and Top-K Accuracy are used in the 18-family sweep. On the representative selected family `moderate__moderate__edgegpu__raspi4__energy_off`, we additionally report Top-K Composite, Pareto-Diverse, and Random-K.

For all scalarized-violation baselines, the per-attribute normalized violation is

$$v_{r,j}(x) = \begin{cases} \dfrac{\max\{0, \tau_{r,j} - a_j(x)\}}{\kappa_{r,j}}, & \text{if } j \text{ is a benefit attribute } (\geq), \\ \dfrac{\max\{0, a_j(x) - \tau_{r,j}\}}{\kappa_{r,j}}, & \text{if } j \text{ is a cost attribute } (\leq), \end{cases} \tag{25}$$

where benefit attributes are accuracies and cost attributes are latencies / energy. The bundled violation of model x on profile r is

$$\phi_{x,r} = \max_{j \in S_r} v_{r,j}(x). \tag{26}$$

We use $\varepsilon = 10^{-12}$ and the following normalization scales:

$$\kappa_{r,j}^{\text{thr}} = \max\{|\tau_{r,j}|, \varepsilon\}, \tag{27}$$

$$\kappa_{r,j}^{\text{mm}} = \max\Big\{ \max_{x' \in \mathcal{M}} a_j(x') - \min_{x' \in \mathcal{M}} a_j(x'), \varepsilon \Big\}, \tag{28}$$

$$\kappa_{r,j}^{\text{iqr}} = \max\Big\{ Q_{0.75}(\{a_j(x')\}_{x' \in \mathcal{M}}) - Q_{0.25}(\{a_j(x')\}_{x' \in \mathcal{M}}), \varepsilon \Big\}. \tag{29}$$

All statistics in (27)–(29) are computed over the full run-specific dataset-filtered archive $\mathcal{M}$, not over a Pareto front, selected subset, or random sub-pool. Unless otherwise noted, the default scalarized normalization is threshold-relative.

Top-K Composite ranks models by

$$s_{\text{comp}}(x) = \frac{1}{|R_{\text{train}}|} \sum_{r \in R_{\text{train}}} \phi_{x,r}, \tag{30}$$

and selects the K models with smallest $s_{\text{comp}}(x)$.

FewForMany uses the same bundled violation (26), defines the current residual portfolio violation by

$$\phi_r(A) = \min_{x \in A} \phi_{x,r}, \qquad \phi_r(\varnothing) = +\infty, \tag{31}$$

and, at each forward step, adds the candidate minimizing

$$\Phi_\omega(A \cup \{x\}) = \frac{\sum_{r \in R_{\text{train}}} \omega_r \min\{\phi_r(A), \phi_{x,r}\}}{\sum_{r \in R_{\text{train}}} \omega_r}, \qquad \omega_r = \begin{cases} 1, & \text{uniform-profile runs,} \\ w_r, & \text{weighted-demand runs.} \end{cases} \tag{32}$$

At each forward step, the deterministic selector scores every remaining model in archive order.

For Pareto-based baselines, let the active objective set be $(\mathrm{acc}, \mathrm{lat}_{h_1}, \mathrm{lat}_{h_2})$ for energy-off families and $(\mathrm{acc}, \mathrm{lat}_{h_1}, \mathrm{lat}_{h_2}, \mathrm{energy})$ for energy-on families. We first convert to benefit-oriented objectives

$$b_j(x) = \begin{cases} a_j(x), & \text{if } j \text{ is accuracy,} \\ -a_j(x), & \text{if } j \text{ is latency or energy,} \end{cases} \tag{33}$$

compute the Pareto front on the unnormalized vectors $b(x)$, and then use min-max normalized benefit vectors

$$\tilde{b}_j(x) = \frac{b_j(x) - \min_{x' \in \mathcal{M}} b_j(x')}{\max_{x' \in \mathcal{M}} b_j(x') - \min_{x' \in \mathcal{M}} b_j(x')} \tag{34}$$

for representative-subset or diversity selection on the front. Distances on the front are always

$$d(x, y) = \|\tilde{b}(x) - \tilde{b}(y)\|_2, \qquad d(x, A) = \min_{z \in A} d(x, z). \tag{35}$$

The min/max statistics in (34) are computed over the full run-specific archive $\mathcal{M}$. If PF denotes the Pareto front, then Pareto-K uses the normalized-front centroid

$$c = \frac{1}{|\mathrm{PF}|} \sum_{y \in \mathrm{PF}} \tilde{b}(y), \tag{36}$$

selects

$$x_1 \in \arg\min_{x \in \mathrm{PF}} \|\tilde{b}(x) - c\|_2, \tag{37}$$

and for $t \geq 2$ selects

$$x_t \in \arg\min_{x \in \mathrm{PF} \setminus A_{t-1}} \frac{1}{|\mathrm{PF}|} \sum_{y \in \mathrm{PF}} \min\{d(y, A_{t-1}), d(y, x)\}, \qquad A_{t-1} = \{x_1, \ldots, x_{t-1}\}. \tag{38}$$

Pareto-Diverse instead selects

$$x_1 \in \arg\max_{x \in \mathrm{PF}} \tilde{b}_{\mathrm{acc}}(x), \tag{39}$$

and for $t \geq 2$ selects

$$x_t \in \arg\max_{x \in \mathrm{PF} \setminus A_{t-1}} d(x, A_{t-1}). \tag{40}$$

If $|\mathrm{PF}| < K$, both Pareto-K and Pareto-Diverse are filled to size K using Top-K Accuracy in stable descending-accuracy order.

D.2 Baseline sanity

These diagnostics are intended to separate structural objective mismatch from implementation failure: some weak baselines satisfy individual constraints frequently yet still fail under bundled feasibility.

Table 14 shows that the baseline failures are structural rather than implementation artifacts. Top-K Accuracy always satisfies the accuracy floor but never either device-latency constraint. Pareto-Diverse achieves nonzero single-attribute pass rates yet zero mean bundled pass rate. Top-K Composite has identical selected-family conditional coverage across K, consistent with later-ranked models adding no new bundled coverage on this family.

D.3 Normalization sensitivity

Table 15 varies only the violation normalization used by FewForMany; Greedy itself is unchanged. Using the notation of Appendix D.1, for a profile $r = (S_r, \tau_r, \sigma_r)$ and a model $x \in \mathcal{M}$, we define

$$v_{r,j}(x) = \frac{\big[\sigma_{r,j}\big(\tau_{r,j} - a_j(x)\big)\big]_+}{\kappa_{r,j}}, \qquad \phi_{x,r} = \max_{j \in S_r} v_{r,j}(x), \tag{41}$$

where $[z]_+ := \max\{z, 0\}$, $\sigma_{r,j} = +1$ for benefit constraints ($\geq$, here accuracy), and $\sigma_{r,j} = -1$ for cost constraints ($\leq$, here latency / energy). We use $\varepsilon = 10^{-12}$ and compare the following three

Table 13: Baseline selection rules and tie-breaking. The full candidate set contains 15,625 models in the selected family. The two front-based methods use the same front and the normalized Euclidean distance in (35).

Method	Score and candidates	Selection and ties
Greedy	Weighted bundled coverage $F(A)$ on the full archive. Uniform runs set $w_r = 1$.	Add the largest marginal gain. Ties go to the smaller archive index.
FewForMany	Residual violation (32) on the full archive with threshold-relative scaling. Weighted runs use $\omega_r = w_r$.	Scan in archive order. Replace the incumbent only for an improvement greater than 10^{-12}. Appendix D.3 compares min–max and IQR scaling.
Pareto-K	Compute dominance on unscaled accuracy, negated latencies and, when active, negated energy. Scale front members using the archive minima and maxima.	Start nearest the centroid and add representatives by (37)–(38). Ties follow lexicographic front order.
Pareto-Diverse	Use the same front and distance as Pareto-K.	Start at highest accuracy, then add the most distant candidate by (39)–(40). Ties follow front order.
Top-K Accuracy	Rank the full archive by decreasing accuracy.	Take the first K in stable order.
Top-K Composite	Rank the full archive by increasing threshold-relative score (30).	Take the first K in stable order.
Random-K	Uniform size-K subsets of the full archive.	Average 30 draws without replacement within each subset. The random stream uses seed $u + K$.

All methods use profile seeds $u = 42, \ldots, 46$. Each deterministic method runs once per split. Stable sorts preserve archive order. Both front-based methods fill any remaining slots by stable decreasing accuracy. The front contains 136 models for the selected family and averages 92.00 ± 39.17 across the 18 families. The selected-family comparison also includes Pareto-Diverse, Top-K Composite and Random-K.

scaling rules:

$$\kappa_{r,j}^{\mathrm{thr}} = \max\{|\tau_{r,j}|, \varepsilon\}, \tag{42}$$

$$\kappa_j^{\mathrm{mm}} = \max\Big\{ \max_{x' \in \mathcal{M}} a_j(x') - \min_{x' \in \mathcal{M}} a_j(x'),\ \varepsilon \Big\}, \tag{43}$$

$$\kappa_j^{\mathrm{iqr}} = \max\Big\{ Q_{0.75}\big(\{a_j(x')\}_{x' \in \mathcal{M}}\big) - Q_{0.25}\big(\{a_j(x')\}_{x' \in \mathcal{M}}\big),\ \varepsilon \Big\}. \tag{44}$$

Here $\mathcal{M}$ is the complete dataset-filtered candidate pool for the selected family in that run. The statistics in (43)–(44) are therefore computed over the full archive, not over the Pareto front, the selected subset, or the sampled training profiles. Unless otherwise noted, the main-text FewForMany baseline uses threshold-relative scaling (42).

Because Greedy does not depend on (42)–(44), Table 15 isolates the normalization sensitivity of the FewForMany composite reduction. The normalization choice matters most at $K = 1$. Under threshold-relative scaling, the selected-family Greedy–FewForMany gap is 0.879 ± 0.031; under robust-IQR scaling, it decreases to 0.014 ± 0.030. By contrast, the $K = 3$ gap stays positive across all three normalizations, and the $K = 5$ gap is essentially zero under all of them. The normalization has its largest effect at the singleton budget. Appendix I varies the aggregation norm and attribute weights as well, choosing the settings on held-out validation profiles.

Table 14: Selected-family baseline sanity diagnostic. For each method and K, the first four metric columns report mean per-selected-model pass rates on coverable test profiles for the accuracy constraint, the two individual device-latency constraints, and the full bundled constraint. The last column reports the resulting portfolio-level conditional coverage.

Method	K	Acc pass	EdgeGPU pass	Raspi4 pass	Bundled pass	Portfolio cond. cov.
Top-K Accuracy	1	1.000	0.000	0.000	0.000	0.000
Top-K Accuracy	3	1.000	0.000	0.000	0.000	0.000
Top-K Accuracy	5	1.000	0.000	0.000	0.000	0.000
Pareto-K	1	0.202	1.000	0.776	0.202	0.326
Pareto-K	3	0.401	0.925	0.592	0.067	0.326
Pareto-K	5	0.440	0.755	0.555	0.040	0.326
Pareto-Diverse	1	1.000	0.000	0.000	0.000	0.000
Pareto-Diverse	3	0.333	0.667	0.667	0.000	0.000
Pareto-Diverse	5	0.600	0.600	0.464	0.000	0.000
Top-K Composite	1	0.000	1.000	1.000	0.000	0.000
Top-K Composite	3	0.000	1.000	1.000	0.000	0.000
Top-K Composite	5	0.000	1.000	1.000	0.000	0.000

Table 15: Normalization sensitivity of Greedy versus FewForMany on the selected family. Numbers are Greedy minus FewForMany conditional coverage (mean $\pm$ standard deviation over five seeds), where FewForMany is rerun using the scaling rules in (42)–(44). The main-text FewForMany baseline uses threshold-relative scaling unless otherwise noted.

Normalization	K	Greedy $-$ FewForMany	Greedy wins / 5
Threshold-relative	1	0.879 ± 0.031	5
Threshold-relative	3	0.128 ± 0.014	5
Threshold-relative	5	0.000 ± 0.000	0
Min-max range	1	0.660 ± 0.069	5
Min-max range	3	0.138 ± 0.040	5
Min-max range	5	0.000 ± 0.000	0
Robust IQR	1	0.014 ± 0.030	2
Robust IQR	3	0.029 ± 0.038	2
Robust IQR	5	0.000 ± 0.000	0

D.4 Generalization to held-out i.i.d. profiles

We next examine how performance changes as the number of sampled training profiles increases. At $K = 3$, Greedy remains at 0.628 ± 0.010 raw coverage and 1.000 ± 0.000 conditional coverage from $n = 50$ to $n = 500$ training profiles. FewForMany is flat in the same regime, staying at 0.556 raw coverage and 0.885 conditional coverage, while Pareto-K remains flat at 0.231 ± 0.016 raw coverage and 0.368 ± 0.021 conditional coverage.

At $K = 5$, Greedy results coincide with $K = 3$ due to saturation. FewForMany follows a similar trend and also reaches perfect conditional coverage throughout. By contrast, Pareto-K remains flat at 0.231 ± 0.016 raw coverage and 0.368 ± 0.021 conditional coverage across n.

Methods that optimize against sampled deployment profiles saturate quickly here, whereas Pareto post-selection is comparatively insensitive to the sampled profile set itself. Appendix I evaluates separate test streams and shifts in the request distribution.

D.5 Runtime on the selected family

Table 16 shows that portfolio selection is computationally lightweight on the offline archive. In our current implementation, plain Greedy is already extremely fast. Lazy Greedy is slightly slower on this problem scale, indicating that heap-maintenance overhead dominates any savings in marginal-gain recomputation for the present archive size and small-K regime. Both Greedy variants select a portfolio in under 10 ms on average at the two measured budgets. When a model is added, its feasibility row can be appended after its attributes are measured. After additions or retirements, rerunning selection on the updated matrix yields a portfolio for the current archive.

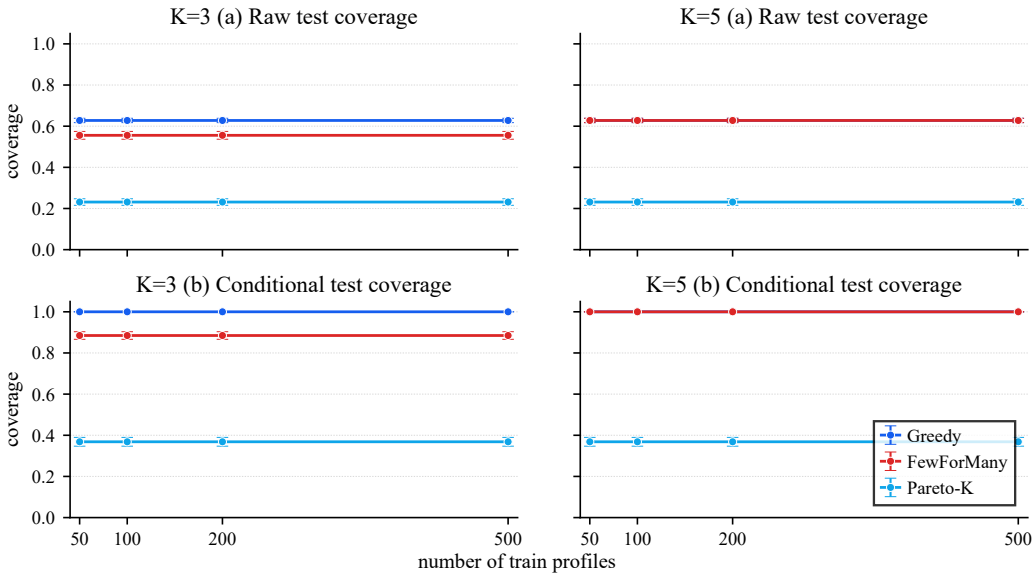


Figure 8: Generalization to held-out profiles under the selected profile family at portfolio sizes $K = 3$ and $K = 5$. Curves show test raw coverage (top) and conditional test coverage (bottom) as the number of sampled training profiles increases. Error bars denote mean $\pm$ standard deviation over five seeds.

Table 16: Wall-clock runtime on the representative selected HW-NAS-Bench family. Numbers are mean $\pm$ standard deviation in milliseconds over five profile seeds.

Method	$K = 3$	$K = 5$
Greedy	**4.93 $\pm$ 0.81**	**5.93 $\pm$ 1.31**
Lazy Greedy	7.83 $\pm$ 1.84	9.13 $\pm$ 2.28
FewForMany	182.34 $\pm$ 23.70	341.53 $\pm$ 105.53
Pareto-K	831.14 $\pm$ 46.25	842.69 $\pm$ 86.23

E Additional Structural and Sweep Diagnostics

E.1 Selected-family structural diagnostics

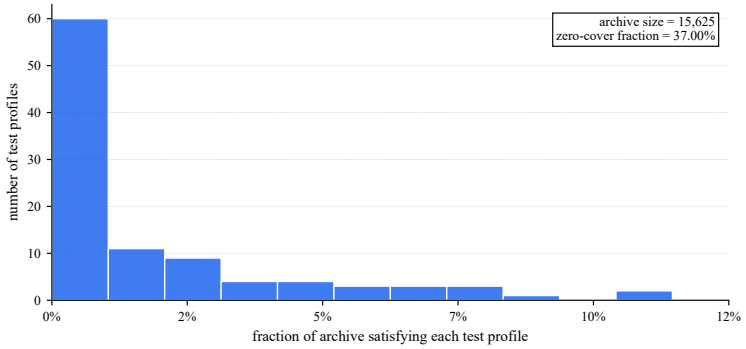


Figure 9: Histogram of test-profile feasibility fractions in the representative test split under the selected family. The horizontal axis reports the number of archive models that satisfy each test profile. The figure highlights the presence of a substantial zero-cover component together with a spread of low- and moderate-feasibility profiles.

Figures 9 and 10 provide two complementary views of problem structure. The feasibility-fraction histogram reveals the distribution of profile sparsity in the sampled test set, while the greedy marginal-

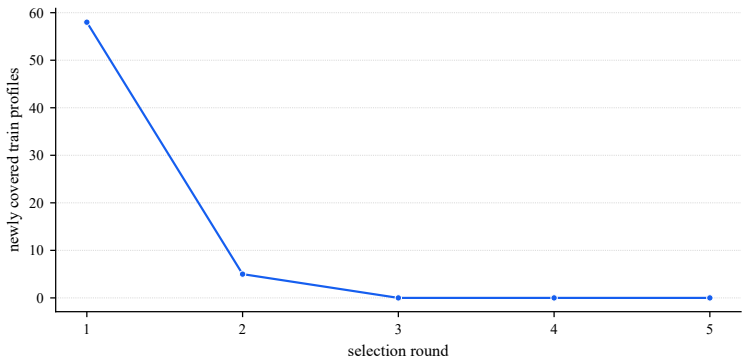


Figure 10: Greedy marginal gain by round in the representative training split under the selected family. Round t denotes adding the t-th model; the plotted value is the marginal coverage gain from $K = t - 1$ to $K = t$. Most of the gain is captured in the first two rounds, followed by rapidly diminishing returns, indicating that the selected family is nontrivial but still exhibits a meaningful small-portfolio regime.

gain curve shows how quickly the coverable subset is absorbed as additional portfolio members are added. Together, they corroborate the claim that the selected family is neither dominated by infeasible profiles nor reducible to a trivial single-model solution.

E.2 Sweep-level structural diagnostics

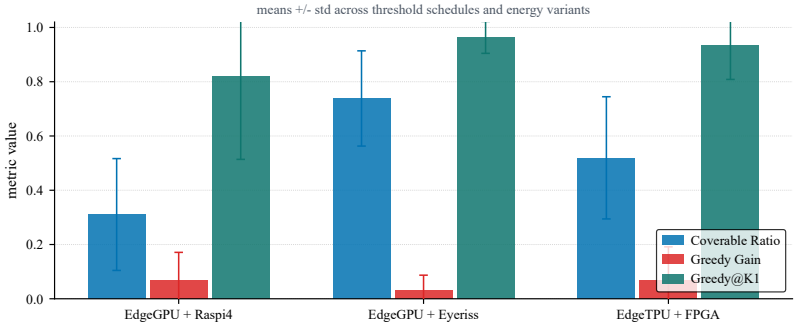


Figure 11: Average structural metrics by device pair, aggregated across threshold schedules and energy variants. Different hardware pairs induce qualitatively different profile structures, affecting both coverability and the observed gain from moving beyond a single deployed model. Error bars denote standard deviation across sweep settings.

Figures 11 and 12 show that problem structure is sensitive not only to the device pair but also to the threshold schedule used in profile generation. This sensitivity helps explain why some deployment-profile families yield trivial or infeasibility-dominated comparisons, whereas others expose meaningful portfolio complementarity.

F Weighted Coverage

F.1 Definition of demand weights

We evaluate three non-uniform demand schemes on sampled profile splits. For each run, profiles are first sampled exactly as in the main experiments, and weights are assigned *after* profile sampling. Thus, the demand schemes do not change the profile generator or the sampling distribution itself; they only reweight the sampled profiles used for optimization and evaluation.

Let $S \in \{\text{train}, \text{test}\}$ denote a split, and let R_S be the corresponding sampled profile set. Each scheme first assigns a nonnegative raw weight $\bar{w}_S(r)$ to each sampled profile $r \in R_S$, and then

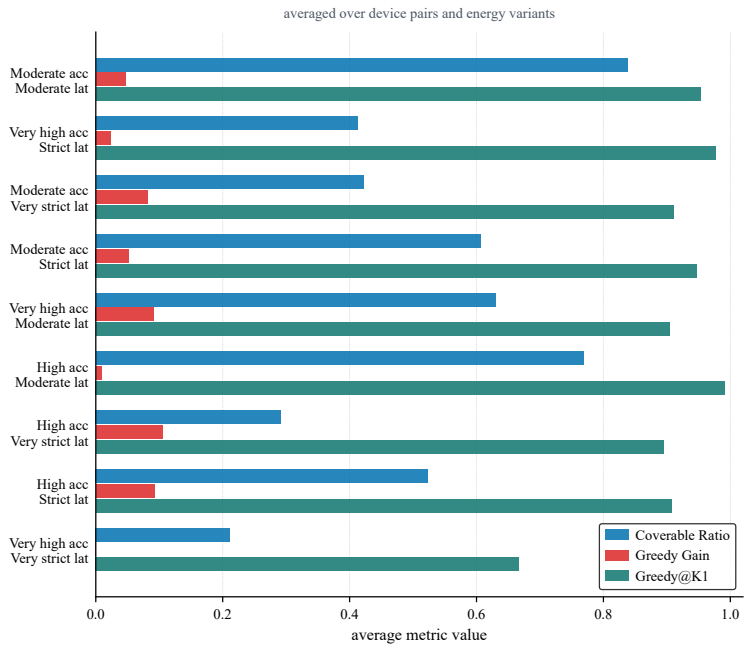


Figure 12: Average structural metrics by threshold combination under an auxiliary 9-schedule diagnostic generator, aggregated across device pairs and energy variants. Unlike the paper-facing 18-family evaluation, which uses only the three schedules Moderate/Moderate, High/Moderate, and Very high/Strict, this auxiliary diagnostic sweeps all 3×3 combinations of accuracy and latency strictness to visualize how coverability and portfolio gain vary with threshold severity. This figure is diagnostic only and is not used in the main 18-family robustness results.

applies split-local mean normalization:

$$w_S(r) = \frac{\bar{w}_S(r)}{\frac{1}{|R_S|}\sum_{r' \in R_S} \bar{w}_S(r')}. \tag{45}$$

This is the exact implementation used in code. Since all weighted metrics divide by the total weight in the relevant split, this normalization is equivalent up to a positive constant to sum-normalization, but we state (45) for exact reproducibility.

For a portfolio A, weighted raw coverage on split S is

$$\mathrm{WRaw}(A; S) = \frac{\sum_{r \in R_S} w_S(r)\, \mathrm{cov}(A, r)}{\sum_{r \in R_S} w_S(r)}. \tag{46}$$

Let

$$R^S_{\mathrm{cov}} = \{r \in R_S : \exists x \in \mathcal{M} \text{ such that } x \models r\} \tag{47}$$

denote the coverable subset of the sampled split with respect to the full dataset-filtered archive $\mathcal{M}$ used in that run. Weighted conditional coverage is then

$$\mathrm{WCond}(A; S) = \frac{\sum_{r \in R^S_{\mathrm{cov}}} w_S(r)\, \mathrm{cov}(A, r)}{\sum_{r \in R^S_{\mathrm{cov}}} w_S(r)}. \tag{48}$$

Thus, infeasible profiles contribute to weighted raw coverage but are excluded from weighted conditional coverage.

Within a fixed run (family, seed, scheme, split), all methods are evaluated with the same weight vector. Greedy and FewForMany are additionally re-optimized under the weighted training objective using w_{train}. By contrast, Pareto-K and Top-K Accuracy retain their original unweighted selection rules and are only re-evaluated using the weighted metrics above.

Business priority (latency-strictness priority). Let $L(r)$ denote the set of active latency attributes in profile r. For each active latency attribute $j \in L(r)$, the profile sampler records its chosen latency quantile $q_j(r)$ in the profile metadata. Define the latency-strictness score

$$\eta(r) = \frac{1}{|L(r)|} \sum_{j \in L(r)} q_j(r). \tag{49}$$

We then assign the tier

$$T(r) = \begin{cases} \text{high}, & \eta(r) \le 0.30, \\ \text{medium}, & 0.30 < \eta(r) \le 0.40, \\ \text{low}, & \eta(r) > 0.40, \end{cases} \tag{50}$$

and raw weights

$$\bar{w}_S(r) = \begin{cases} 10, & T(r) = \text{high}, \\ 5, & T(r) = \text{medium}, \\ 1, & T(r) = \text{low}. \end{cases} \tag{51}$$

This scheme is deterministic conditional on the sampled profiles. Only active latency quantiles enter the tier rule; accuracy and energy do not.

Hard-profile upweighting. For a sampled profile r, let

$$f(r) = \frac{|e_r|}{|\mathcal{M}|}, \qquad e_r = \{x \in \mathcal{M} : x \models r\}, \tag{52}$$

where $\mathcal{M}$ is the full dataset-filtered archive for that run. We use the continuous inverse-feasibility rule

$$\bar{w}_S(r) = \frac{1}{\max\{f(r), \varepsilon\}}, \qquad \varepsilon = \frac{1}{|\mathcal{M}|}. \tag{53}$$

Hence there is no bucketing, no additional exponent, and no explicit cap. If $f(r) = 0$, then $\bar{w}_S(r) = |\mathcal{M}|$ before normalization, so infeasible sampled profiles receive the largest raw weight under this scheme. They are nevertheless excluded from (48) because the conditional denominator is restricted to R^S_{cov}.

Zipf demand. For each split S, we draw an independent random permutation

$$\pi_S : \{1, \dots, |R_S|\} \to R_S$$

of the sampled profiles in that split and assign raw weights by Zipf rank:

$$\bar{w}_S\big(\pi_S(k)\big) = k^{-s}, \qquad k = 1, \dots, |R_S|, \qquad s = 1.2. \tag{54}$$

Thus, rank is not determined by a profile attribute or demand class; it is induced by a split-specific random permutation. Train and test splits are ranked independently. For run seed u, the extra RNG used by Zipf demand is

$$\text{train seed} = 100u + 2, \qquad \text{test seed} = 100u + 52,$$

corresponding to the scheme index of Zipf demand in the experiment loop.

The business-priority and hard-profile schemes introduce no additional randomness beyond the sampled profiles themselves. For Zipf demand, the random permutation is the only extra source of randomness.

G Noise Robustness

We study robustness to archive remeasurement noise on the representative selected family `moderate__moderate__edgegpu__raspi4__energy_off`. This family uses one accuracy attribute and two latency attributes, so the noise model perturbs only these three quantities. For each run, policy, portfolio size K, and noise level, we evaluate fixed-portfolio coverage under $B = 100$ independent noisy remeasurements and evaluate reselection stability under $T = 20$ independent noisy remeasurements.

Table 17: Selected-family weighted-demand summary for the representative `moderate__moderate__edgegpu__raspi4__energy_off` family. Entries are mean ± standard deviation across five seeds.

Scheme	Method	K	Weighted raw	Weighted conditional	Unweighted conditional	Hardest bucket
Business priority	Greedy	1	0.232 ± 0.050	0.823 ± 0.036	0.872 ± 0.014	0.584 ± 0.115
Business priority	Greedy	3	0.282 ± 0.063	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Business priority	Greedy	5	0.282 ± 0.063	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Business priority	FewForMany	1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Business priority	FewForMany	3	0.232 ± 0.050	0.823 ± 0.036	0.872 ± 0.014	0.584 ± 0.115
Business priority	FewForMany	5	0.282 ± 0.063	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Business priority	Pareto-K	1	0.039 ± 0.006	0.145 ± 0.035	0.326 ± 0.048	0.000 ± 0.000
Business priority	Pareto-K	3	0.039 ± 0.006	0.145 ± 0.035	0.326 ± 0.048	0.000 ± 0.000
Business priority	Pareto-K	5	0.039 ± 0.006	0.145 ± 0.035	0.326 ± 0.048	0.000 ± 0.000
Business priority	Top-K Accuracy	1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Business priority	Top-K Accuracy	3	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Business priority	Top-K Accuracy	5	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Hard-profile upweighting	Greedy	1	0.044 ± 0.015	0.927 ± 0.028	0.872 ± 0.014	0.584 ± 0.115
Hard-profile upweighting	Greedy	3	0.047 ± 0.015	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Hard-profile upweighting	Greedy	5	0.047 ± 0.015	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Hard-profile upweighting	FewForMany	1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Hard-profile upweighting	FewForMany	3	0.044 ± 0.015	0.927 ± 0.028	0.872 ± 0.014	0.584 ± 0.115
Hard-profile upweighting	FewForMany	5	0.047 ± 0.015	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Hard-profile upweighting	Pareto-K	1	0.001 ± 0.000	0.016 ± 0.008	0.326 ± 0.048	0.000 ± 0.000
Hard-profile upweighting	Pareto-K	3	0.001 ± 0.000	0.016 ± 0.008	0.326 ± 0.048	0.000 ± 0.000
Hard-profile upweighting	Pareto-K	5	0.001 ± 0.000	0.016 ± 0.008	0.326 ± 0.048	0.000 ± 0.000
Hard-profile upweighting	Top-K Accuracy	1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Hard-profile upweighting	Top-K Accuracy	3	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Hard-profile upweighting	Top-K Accuracy	5	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Zipf demand	Greedy	1	0.585 ± 0.149	0.893 ± 0.071	0.865 ± 0.018	0.518 ± 0.141
Zipf demand	Greedy	3	0.657 ± 0.163	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Zipf demand	Greedy	5	0.657 ± 0.163	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Zipf demand	FewForMany	1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Zipf demand	FewForMany	3	0.591 ± 0.157	0.901 ± 0.077	0.872 ± 0.014	0.584 ± 0.115
Zipf demand	FewForMany	5	0.657 ± 0.163	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Zipf demand	Pareto-K	1	0.284 ± 0.174	0.428 ± 0.197	0.326 ± 0.048	0.000 ± 0.000
Zipf demand	Pareto-K	3	0.284 ± 0.174	0.428 ± 0.197	0.326 ± 0.048	0.000 ± 0.000
Zipf demand	Pareto-K	5	0.284 ± 0.174	0.428 ± 0.197	0.326 ± 0.048	0.000 ± 0.000
Zipf demand	Top-K Accuracy	1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Zipf demand	Top-K Accuracy	3	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Zipf demand	Top-K Accuracy	5	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000

Noise model. Let $a_{\mathrm{acc}}(x)$ denote the clean accuracy of model x, and let $a_j(x)$ denote a clean latency attribute for device j. For Monte Carlo replicate b, the noisy accuracy is

$$\tilde{a}^{(b)}_{\mathrm{acc}}(x) = a_{\mathrm{acc}}(x) + \epsilon^{(b)}_{\mathrm{acc}}(x), \qquad \epsilon^{(b)}_{\mathrm{acc}}(x) \sim \mathcal{N}(0, \sigma^2_{\mathrm{acc}}). \tag{55}$$

Accuracy noise is applied in absolute accuracy units on $[0, 1]$ and is *not* clipped back to $[0, 1]$.

For each latency attribute j, the noisy value is

$$\tilde{a}^{(b)}_j(x) = \begin{cases} a_j(x)\, \max\left\{1 + \epsilon^{(b)}_j(x), 10^{-6}\right\}, & a_j(x) > 0, \\ a_j(x), & a_j(x) \le 0, \end{cases} \qquad \epsilon^{(b)}_j(x) \sim \mathcal{N}(0, \sigma^2_{\mathrm{lat}}). \tag{56}$$

Thus, latency noise is multiplicative and uses a positivity floor of 10^{-6} on the multiplicative factor. All Gaussian draws are independent across attributes, models, and Monte Carlo replicates; in particular, the two latency devices do not share a common noise term.

We use four noise levels:

$$\begin{array}{lll} \textsc{Clean}: & \sigma_{\mathrm{acc}} = 0, & \sigma_{\mathrm{lat}} = 0, \\ \textsc{Low}: & \sigma_{\mathrm{acc}} = 0.001, & \sigma_{\mathrm{lat}} = 0.02, \\ \textsc{Medium}: & \sigma_{\mathrm{acc}} = 0.0025, & \sigma_{\mathrm{lat}} = 0.05, \\ \textsc{High}: & \sigma_{\mathrm{acc}} = 0.005, & \sigma_{\mathrm{lat}} = 0.10. \end{array} \tag{57}$$

Equivalently, the accuracy standard deviations correspond to 0, 0.1, 0.25, and 0.5 percentage points.

Expected and chance conditional coverage. Let R_{test} be the sampled test profiles for the run, and define the clean-archive coverable subset

$$R_{\mathrm{cov}} = \{r \in R_{\mathrm{test}} : \exists x \in \mathcal{M} \text{ such that } x \models r\}, \tag{58}$$

Table 18: 18-family aggregate under non-uniform demand. Entries report family-mean performance; weighted conditional coverage additionally reports standard deviation across families.

Scheme	Method	K	Family-mean weighted conditional	Family-mean unweighted conditional	Family-mean hardest bucket
Business priority	Greedy	1	0.963 ± 0.069	0.975	0.931
Business priority	Greedy	3	1.000 ± 0.000	1.000	1.000
Business priority	Greedy	5	1.000 ± 0.000	1.000	1.000
Business priority	FewForMany	1	0.420 ± 0.475	0.430	0.415
Business priority	FewForMany	3	0.782 ± 0.352	0.804	0.727
Business priority	FewForMany	5	1.000 ± 0.000	1.000	1.000
Business priority	Pareto-K	1	0.053 ± 0.078	0.147	0.002
Business priority	Pareto-K	3	0.057 ± 0.077	0.164	0.002
Business priority	Pareto-K	5	0.150 ± 0.215	0.259	0.092
Business priority	Top-K Accuracy	1	0.000 ± 0.000	0.000	0.000
Business priority	Top-K Accuracy	3	0.000 ± 0.000	0.000	0.000
Business priority	Top-K Accuracy	5	0.000 ± 0.000	0.000	0.000
Hard-profile upweighting	Greedy	1	0.964 ± 0.080	0.974	0.930
Hard-profile upweighting	Greedy	3	1.000 ± 0.000	1.000	1.000
Hard-profile upweighting	Greedy	5	1.000 ± 0.000	1.000	1.000
Hard-profile upweighting	FewForMany	1	0.332 ± 0.455	0.387	0.326
Hard-profile upweighting	FewForMany	3	0.765 ± 0.388	0.796	0.715
Hard-profile upweighting	FewForMany	5	0.889 ± 0.314	0.889	0.889
Hard-profile upweighting	Pareto-K	1	0.011 ± 0.018	0.147	0.002
Hard-profile upweighting	Pareto-K	3	0.012 ± 0.018	0.164	0.002
Hard-profile upweighting	Pareto-K	5	0.063 ± 0.115	0.259	0.092
Hard-profile upweighting	Top-K Accuracy	1	0.000 ± 0.000	0.000	0.000
Hard-profile upweighting	Top-K Accuracy	3	0.000 ± 0.000	0.000	0.000
Hard-profile upweighting	Top-K Accuracy	5	0.000 ± 0.000	0.000	0.000
Zipf demand	Greedy	1	0.979 ± 0.040	0.976	0.933
Zipf demand	Greedy	3	1.000 ± 0.000	1.000	1.000
Zipf demand	Greedy	5	1.000 ± 0.000	1.000	1.000
Zipf demand	FewForMany	1	0.440 ± 0.472	0.442	0.424
Zipf demand	FewForMany	3	0.890 ± 0.200	0.883	0.827
Zipf demand	FewForMany	5	0.989 ± 0.046	0.989	0.989
Zipf demand	Pareto-K	1	0.172 ± 0.209	0.147	0.002
Zipf demand	Pareto-K	3	0.194 ± 0.206	0.164	0.002
Zipf demand	Pareto-K	5	0.286 ± 0.293	0.259	0.092
Zipf demand	Top-K Accuracy	1	0.000 ± 0.000	0.000	0.000
Zipf demand	Top-K Accuracy	3	0.000 ± 0.000	0.000	0.000
Zipf demand	Top-K Accuracy	5	0.000 ± 0.000	0.000	0.000

Table 19: Greedy versus FewForMany win counts under non-uniform demand on the 18-family sweep. Positive gaps favor Greedy.

Scheme	K	Greedy − Few gap	Greedy wins	Ties
Business priority	1	0.543 ± 0.461	11	6
Business priority	3	0.218 ± 0.352	7	11
Business priority	5	0.000 ± 0.000	0	18
Hard-profile upweighting	1	0.631 ± 0.452	12	6
Hard-profile upweighting	3	0.235 ± 0.388	7	11
Hard-profile upweighting	5	0.111 ± 0.314	2	16
Zipf demand	1	0.539 ± 0.466	12	6
Zipf demand	3	0.110 ± 0.200	7	11
Zipf demand	5	0.011 ± 0.046	1	17

where $\mathcal{M}$ is the clean archive. For a fixed selected portfolio A, define the empirical noisy coverage probability of profile r by

$$p_r(A) = \frac{1}{B} \sum_{b=1}^{B} \mathbf{1}[\mathrm{cov}_{\xi_b}(A, r) = 1], \tag{59}$$

where $\mathrm{cov}_{\xi_b}(A, r)$ evaluates bundled coverage of r by the fixed portfolio A on the b-th noisy remeasurement of the archive.

We report expected raw coverage

$$\mathrm{ExpRaw}(A) = \frac{1}{|R_{\mathrm{test}}|} \sum_{r \in R_{\mathrm{test}}} p_r(A), \tag{60}$$

expected conditional coverage

$$\mathrm{ExpCond}(A) = \frac{1}{|R_{\mathrm{cov}}|} \sum_{r \in R_{\mathrm{cov}}} p_r(A), \tag{61}$$

and chance conditional coverage

$$\mathrm{ChanceCond}(A) = \frac{1}{|R_{\mathrm{cov}}|} \sum_{r \in R_{\mathrm{cov}}} \mathbf{1}[p_r(A) \geq q], \qquad q = 0.90. \tag{62}$$

Hence "Chance cond." reports the fraction of clean-coverable test profiles whose empirical coverage probability under noisy remeasurement is at least 0.90. Both (61) and (62) use the portfolio selected once on the clean archive; there is no noisy reselection in these two metrics.

Reselection stability. Let A^{nom} denote the portfolio selected on the clean archive for a fixed policy and budget K. For each of $T = 20$ independent noisy remeasurements, we reselect a portfolio $\tilde{A}^{(t)}$ on the noisy archive using the *same* policy and the *same* K. Stability is defined as the mean Jaccard similarity between the nominal portfolio and the noisy reselected portfolios:

$$\mathrm{Stability}(A^{\mathrm{nom}}) = \frac{1}{T} \sum_{t=1}^{T} J\left(A^{\mathrm{nom}}, \tilde{A}^{(t)}\right), \qquad J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{63}$$

Thus, "Stability" is *not* a pairwise Jaccard average among noisy portfolios. Under the clean noise level, noisy reselection is identical to nominal reselection, so the clean stability is 1.

Margin-aware policy. The margin-aware policy tightens the *training* profile thresholds before selection and then evaluates the resulting portfolio against the original deployment test profiles under noisy remeasurement. For an accuracy threshold τ_{acc} and latency threshold τ_{lat}, the tightened internal thresholds are

$$\tau'_{\mathrm{acc}} = \tau_{\mathrm{acc}} + m_{\mathrm{acc}}, \tag{64}$$

$$\tau'_{\mathrm{lat}} = \tau_{\mathrm{lat}} \max\{1 - m_{\mathrm{lat}}, 10^{-12}\}. \tag{65}$$

The margins depend only on the noise level:

$$\begin{array}{lll} \textsc{Clean}: & m_{\mathrm{acc}} = 0, & m_{\mathrm{lat}} = 0, \\ \textsc{Low}: & m_{\mathrm{acc}} = 0.001, & m_{\mathrm{lat}} = 0.01, \\ \textsc{Medium}: & m_{\mathrm{acc}} = 0.0025, & m_{\mathrm{lat}} = 0.03, \\ \textsc{High}: & m_{\mathrm{acc}} = 0.005, & m_{\mathrm{lat}} = 0.05. \end{array} \tag{66}$$

These margins are fixed across seeds and portfolio sizes for a given noise level.

All Monte Carlo replicates use fresh independent draws. In the implementation, expected/chance metrics and stability use deterministic RNG schedules derived from the run seed, noise level, policy, and budget K.

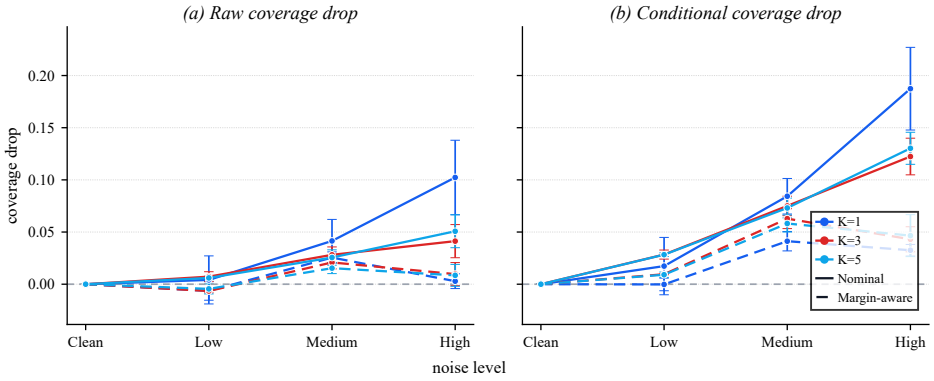


Figure 13: Coverage-drop view of the selected-family noise-robustness study for Greedy-selected portfolios. Solid lines denote nominal selection and dashed lines denote margin-aware selection. The plotted quantities are drops in *expected* coverage relative to the corresponding clean baseline of the same policy: $\mathrm{raw_drop} = \mathrm{ExpRaw}_{\mathrm{clean}} - \mathrm{ExpRaw}$ and $\mathrm{cond_drop} = \mathrm{ExpCond}_{\mathrm{clean}} - \mathrm{ExpCond}$.

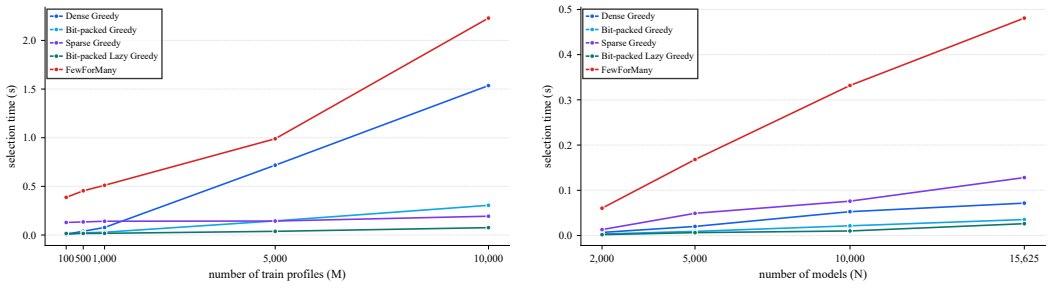
The selected-family noise study shows three consistent patterns. First, medium noise does not overturn the main $K \geq 3$ conclusions: nominal Greedy still achieves expected/chance conditional

Table 20: Attribute-noise robustness for Greedy-selected portfolios on the representative selected family. "Chance cond." reports the fraction of profiles whose coverage probability under noisy remeasurement exceeds the evaluation threshold used in the experiment; "Stability" reports the Jaccard similarity of portfolios under noisy reselection.

Policy	Noise	K	Exp. cond.	Chance cond.	Stability
Nominal	Clean	1	0.884 ± 0.019	0.884 ± 0.019	1.000 ± 0.000
Nominal	Clean	3	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Nominal	Clean	5	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Nominal	Low	1	0.867 ± 0.023	0.799 ± 0.079	0.190 ± 0.058
Nominal	Low	3	0.972 ± 0.004	0.893 ± 0.030	0.444 ± 0.045
Nominal	Low	5	0.972 ± 0.002	0.901 ± 0.027	0.559 ± 0.028
Nominal	Medium	1	0.800 ± 0.014	0.616 ± 0.114	0.010 ± 0.020
Nominal	Medium	3	0.925 ± 0.009	0.771 ± 0.008	0.212 ± 0.017
Nominal	Medium	5	0.927 ± 0.009	0.771 ± 0.008	0.433 ± 0.010
Nominal	High	1	0.697 ± 0.027	0.236 ± 0.065	0.000 ± 0.000
Nominal	High	3	0.878 ± 0.018	0.651 ± 0.033	0.192 ± 0.002
Nominal	High	5	0.870 ± 0.015	0.631 ± 0.054	0.437 ± 0.013
Margin-aware	Clean	1	0.884 ± 0.019	0.884 ± 0.019	1.000 ± 0.000
Margin-aware	Clean	3	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Margin-aware	Clean	5	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Margin-aware	Low	1	0.885 ± 0.014	0.866 ± 0.035	0.270 ± 0.117
Margin-aware	Low	3	0.990 ± 0.000	0.953 ± 0.025	0.445 ± 0.045
Margin-aware	Low	5	0.991 ± 0.003	0.969 ± 0.026	0.571 ± 0.025
Margin-aware	Medium	1	0.853 ± 0.018	0.716 ± 0.009	0.080 ± 0.117
Margin-aware	Medium	3	0.937 ± 0.010	0.830 ± 0.016	0.269 ± 0.020
Margin-aware	Medium	5	0.942 ± 0.008	0.822 ± 0.032	0.461 ± 0.026
Margin-aware	High	1	0.861 ± 0.017	0.703 ± 0.022	0.020 ± 0.024
Margin-aware	High	3	0.851 ± 0.021	0.697 ± 0.021	0.444 ± 0.015
Margin-aware	High	5	0.847 ± 0.014	0.676 ± 0.039	0.614 ± 0.023

coverage $0.925/0.771$ at $K = 3$ and $0.927/0.771$ at $K = 5$. Second, singleton portfolios are much more brittle under severe noise: nominal $K = 1$ drops to 0.697 expected and 0.236 chance conditional coverage under high noise. Third, the margin-aware policy improves chance coverage at low-to-medium noise and substantially improves reselection stability under severe noise, although for $K \geq 3$ at high noise this comes with a mild coverage penalty rather than a uniform gain.

H Scaling and Memory



(a) Selection runtime vs. number of profiles M.

(b) Selection runtime vs. archive size N.

Figure 14: Selection-time scaling for dense, bit-packed, and sparse Greedy implementations, together with bit-packed Lazy Greedy and FewForMany. Lazy Greedy only becomes beneficial once the number of profiles is sufficiently large.

Runtime scaling is quantified along both the number of sampled profiles M and the archive size N. The most relevant implementation conclusion is that bit-packed Lazy Greedy first beats bit-packed Greedy at about $M = 500$, which explains why lazy evaluation is not beneficial on the

Table 21: Representative maintained feasibility-storage footprint at a large setting on the selected family ($K = 5$). The reported memory is the maintained encoded feasibility footprint rather than full OS-level peak RSS.

Storage scheme	Build+encode (s)	Storage (MiB)	Same portfolio	Same coverage
Dense bool	1.147	74.51	1	1
Bit-packed	1.182	9.31	1	1
Sparse incidence	1.190	2.82	1	1

smaller benchmark instances emphasized in the main text. For maintained feasibility storage at a representative large setting, bit-packed and sparse-incidence encodings reduce the footprint from 74.51 MiB to 9.31 MiB and 2.82 MiB, respectively, while preserving the selected portfolio and achieved coverage exactly.

I Additional Sensitivity and Ablation Studies

Figure 15 compares portfolio size, scalarization settings, the accuracy dataset and the distribution of deployment requests. Panels (a)–(c) report held-out conditional coverage or comparisons derived from it. Panel (d) uses reduced instances small enough to solve both portfolio objectives exactly.

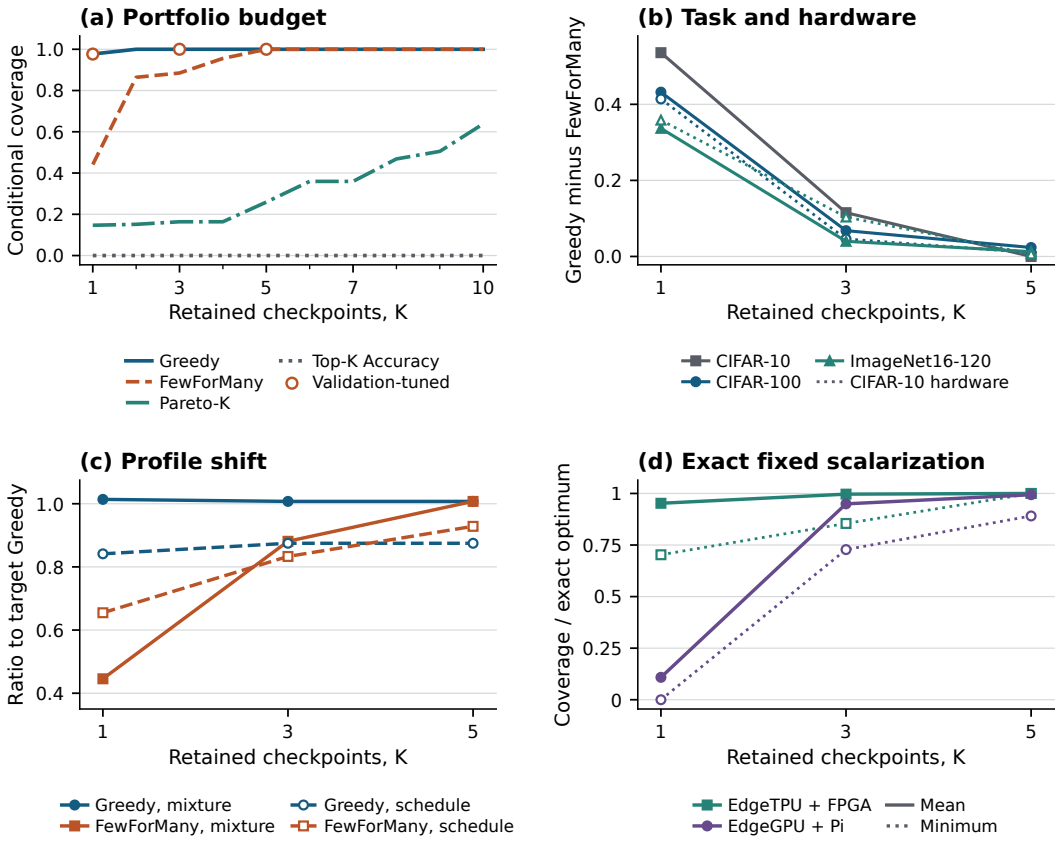


Figure 15: Portfolio sensitivity. (a) Mean conditional test coverage across 18 CIFAR-10 families. Validation-tuned scalarization is evaluated at $K = 1, 3, 5$. (b) Greedy minus fixed FewForMany coverage with task-specific or fixed CIFAR-10 hardware measurements. (c) Source-trained coverage divided by target-trained Greedy coverage on the same shifted test profiles. (d) Mean and minimum coverage of the exact fixed-scalarization solution relative to the maximum-coverage optimum, over 60 reduced instances per device pair and budget.

Portfolio budget and validation tuning (panel a). For the 18 CIFAR-10 families, the mean Greedy–FewForMany gap in conditional coverage falls from 0.536 at $K = 1$ to 0.136, 0.115 and 0.044 at $K = 2, 3, 4$. Both methods cover every coverable test profile from $K = 5$ through $K = 10$. To separate the effect of a fixed scalarization from the choice of objective, we also select portfolios on 100 training profiles using the score

$$\phi_{x,r}^{(p,\alpha)} = \left\| (\alpha_j v_{r,j}(x))_{j \in S_r} \right\|_p, \qquad p \in \{1, 2, \infty\}.$$

We cross the three norms with threshold-relative, min–max and interquartile-range scaling. We either weight all attributes equally or multiply one attribute's weight by 2 or 5. This gives 63 settings without energy and 81 with energy. For each family, seed u and budget, we select the setting with the highest conditional coverage on 100 separate validation profiles (seed $u + 10007$), then evaluate the selected portfolio on the test profiles. At $K = 1$, tuned scalarization reaches 0.9767 mean conditional coverage and Greedy reaches 0.9770 without a validation set. The two tie in 15 of 18 families at $K = 1$ and in all 18 at $K = 3$ (Figure 2a).

Task accuracy and hardware measurements (panel b). We evaluate the same NAS-Bench-201 architectures on CIFAR-100 and ImageNet16-120. The generator uses the percentile levels of the CIFAR-10 accuracy and EdgeGPU latency grids in Appendix B (empirical midranks rounded to the nearest percentile), applying them to the corresponding target attributes. With each task's hardware records, mean singleton gains over fixed FewForMany are 0.432 and 0.337, respectively. For the nine families below each task's median coverable ratio, they rise to 0.766 and 0.521. To isolate the change in accuracy, we match architecture identifiers and keep CIFAR-10 hardware costs on both tasks. The resulting singleton gains are 0.414 and 0.358. Thus the concentration of singleton gains can be seen across tasks within the same architecture space. On the six EdgeTPU–FPGA families, both methods reach full conditional coverage at $K = 3, 5$ on CIFAR-100. On ImageNet16-120 the two high/moderate families retain a gap at $K = 3$, and all six tie at $K = 5$.

Disjoint test streams and shifted requests (panel c). We first keep the CIFAR-10 profile generator fixed and pair training seed u with test seed $100000 + u$. The 18-family gaps at $K = 1, 3, 5$ are 0.539, 0.110 and 0.000. We then hold the source-trained portfolio fixed while changing the test distribution. Each plotted value averages per-run coverage ratios relative to Greedy fitted on an independent sample from the target distribution. This reference is a learned portfolio, so a ratio can exceed one. Changing the easy, medium and hard mixture from $(0.34, 0.33, 0.33)$ to $(0.10, 0.20, 0.70)$ gives 90 family–seed pairs and Greedy ratios of 1.014, 1.007 and 1.007. A second experiment tightens the threshold schedule by one level across 60 family–seed pairs. Greedy retains more coverage than fixed FewForMany at $K = 1$ (ratios 0.841 and 0.655). At $K = 5$ the ordering reverses (0.875 and 0.928). The choice of portfolio can therefore matter under tighter thresholds even when both methods saturate on the source distribution.

Exact fixed scalarization (panel d). For the moderate, energy-off EdgeGPU–Raspi4 and EdgeTPU–FPGA families, we solve the fixed scalarized portfolio objective as a p-median mixed-integer program,

$$\min_{\substack{A \subseteq \mathcal{M}' \\ |A| = K}} \sum_{r \in \mathcal{R}'} \min_{x \in A} \phi_{x,r},$$

using the default violation in (26). We measure the resulting portfolio's bundled coverage against the exact maximum-coverage optimum on the same instance (Appendix C.6). Each device pair contributes 60 instances at each of three budgets, for 360 instance–budget evaluations in total. Greedy reaches the coverage optimum in every evaluation. At $K = 1$, exact fixed scalarization reaches a mean ratio of 0.109 on EdgeGPU–Raspi4 (minimum 0.000) and 0.952 on EdgeTPU–FPGA (minimum 0.703). At $K = 3$ the means rise to 0.950 and 0.997, and at $K = 5$ to 0.994 and 1.000. Solving the fixed violation objective exactly can therefore select a different portfolio from one that maximizes bundled coverage, particularly at the singleton budget.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contributions—(i) a distributional formulation of the portfolio-selection problem, (ii) theoretical results including the approximation guarantee (Theorem 3) and the generalization bound (Theorem 4, Corollary 5), and (iii) empirical evaluation on the benchmark described in Section 6. Each claim is substantiated in the corresponding section (Sections 3–6), and the scope (a finite archive of candidate models, coverage-style objectives, profile generator of Appendix B.2) is stated explicitly.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the dedicated Limitations paragraph of Section 7 and in Appendix A.4, where we explicitly note that (i) Theorem 4 is a worst-case union bound that is loose at our sample size n=100 (uniform deviation 0.495 at $K = 5$), (ii) the benchmark profile generator draws thresholds from a finite grid (Appendix B.2, Tables 5–6) which bounds the diversity of deployment scenarios tested, and (iii) results assume a finite archive and access to offline evaluations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All formal statements (Propositions 2 and 6, Theorems 1, 3–4, Corollary 5) list their assumptions inline, and proofs or references are provided in the main text and Appendix A with cross-references. Short explanations appear in the main text alongside these results to build intuition.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 6 and Appendix B describe the archive construction, profile generator (including the exact threshold grids in Tables 5–6), train/test sample sizes, evaluation metrics, baselines, and random seeds. Appendix D.1 lists the baseline settings.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The repository at `https://github.com/jicheng9617/F4MwCons` provides commands to obtain the public benchmark inputs, generate the profiles, and reproduce the main tables and figures with pinned dependencies.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

Justification: Section 6.1 specifies the train/test split (n_{train}=n_{test}=100, with a sample-size sweep in Appendix D.4), portfolio size K, coverage threshold τ, and seeds. Appendix D.1 specifies the baseline settings, and Appendix I gives the scalarization grid and validation protocol.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Selected-family tables and figures report mean $\pm$ one standard deviation across 5 profile seeds, as stated in the captions of Table 8 and Figures 3–4; Table 2 summarizes variation across families. Figure 8 in Appendix D.4 additionally reports across-seed standard deviations for the sample-size sweep.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B.4 reports that experiments were run on a single machine with a standard multi-core CPU (no GPU required), with 32 GB RAM. Memory usage and per-experiment runtimes are listed in Appendices C, D and H. There is no estimate of compute spent on preliminary and failed runs.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is algorithmic and uses only synthetic profiles and a derived archive of evaluation metrics; no human subjects, no personal data, and no sensitive attributes are involved. We have reviewed the Code of Ethics and conform to it in all respects.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader-impact discussion in Section 7 notes that reliable model-portfolio selection can reduce maintenance costs at deployment (positive impact), while acknowledging that the framework could amplify imbalances present in the profile distribution, which we discuss as a limitation for infrequent requests.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper describes selection algorithms and a benchmark of aggregate evaluation metrics over an archive of candidate models; no generative models, no scraped data, and no user-facing artifacts with foreseeable misuse risk are released.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: The third-party benchmark assets (HW-NAS-Bench and NAS-Bench-201) are cited in the main text. Their public implementations have MIT licenses. Their terms of use permit research reuse and are respected.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The repository at `https://github.com/jicheng9617/F4MwCons` documents the profile generator and portfolio selectors, includes configuration files and a worked example, and provides the original code under the MIT License.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects; all data are synthetic or derived from offline model evaluations.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, so IRB approval is not applicable.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used as a component of the core methodology, theory, or experiments. Any use of LLMs for writing and editing support did not provide empirical measurements.