GUIHUB: PERSONALIZING GUI AGENTS VIA CAPABILITY-DRIVEN LORA COMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graphical user interface (GUI) agents must adapt to task distributions that vary across users. Independently trained low-rank adaptation (LoRA) adapters offer specialized capabilities, but reusing them for a target user with only a few examples remains challenging. We present **GUIHub**, an asynchronous framework for sharing and composing LoRA adapters. Our key idea is to connect user demand with measured adapter capabilities through shared GUI tasks. GUIHub builds a shared capability space from adapter behavior on these tasks. Semantic relevance between local support examples and the shared tasks specifies user demand in this space, guiding adapter selection and weighting without local action labels. The resulting composition is reused across requests, while new adapters join the same space without retraining existing adapters. We evaluate personalization, generalization, mixed-task composition, and Hub expansion on mobile GUI trajectories from FedMABench. GUIHub improves offline Step Success Rate by up to 5.61 pp over the strongest baseline on in-hub and held-out clients. With prebuilt Hub assets, GUIHub achieves a 2.08× speedup over the fastest baseline in recorded setup-plus-execution time on Qwen3.5-4B. Expanding the Hub from 8 to 32 adapters improves Step Success Rate by 6.16–7.61 pp across all three backbones on the same 32 target clients.

1 INTRODUCTION

Graphical user interface (GUI) agents translate natural-language instructions and screen observations into actions across mobile, web, and desktop environments (Deng et al., 2023; Rawles et al., 2023; Zhou et al., 2024; Xie et al., 2024; Zhang et al., 2025b;a; Rawles et al., 2025). Recent models have improved GUI grounding and action prediction (Hong et al., 2024; Cheng et al., 2024; Lin et al., 2025; Wu et al., 2025; Qin et al., 2025), yet users differ in the applications they use and the mix of tasks they perform (Nie et al., 2026; Wang et al., 2025; 2026). Consequently, an agent that performs well on average may still be poorly matched to a particular user's task distribution. Low-rank adaptation (LoRA) lets users or organizations with local data specialize a shared backbone efficiently (Hu et al., 2022). As these specialized adapters accumulate, organizing and reusing their capabilities becomes a practical deployment problem.

Building on prior work in adapter composition and routing (Huang et al., 2024; Zhao et al., 2024; Lee et al., 2026), we introduce **GUIHub** (Figure 1), an evolving Hub of interface-compatible GUI adapters sharing a common backbone. Contributors independently submit new or updated adapters. Given a small representative local support set (LSS), GUIHub selects and composes adapters for reuse across subsequent GUI tasks. This enables existing users to benefit from others' adapters and unseen users to adapt without training their own. Unlike FedAvg-style federated training (McMahan et al., 2017), GUIHub accepts independently trained adapters without synchronized optimization rounds or retraining existing adapters.

This setting raises a central question: *How can we select and compose GUI adapters from a growing Hub to match a user's task distribution, given only a small local support set?* Personalized reuse requires both identifying the GUI tasks relevant to a user and estimating how well each adapter can perform them. Our key idea is to use shared GUI tasks as a bridge between user demand and adapter capabilities. By linking the local support set to these tasks through semantic relevance, we can select and compose adapters based on their measured behavior on tasks relevant to the user.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

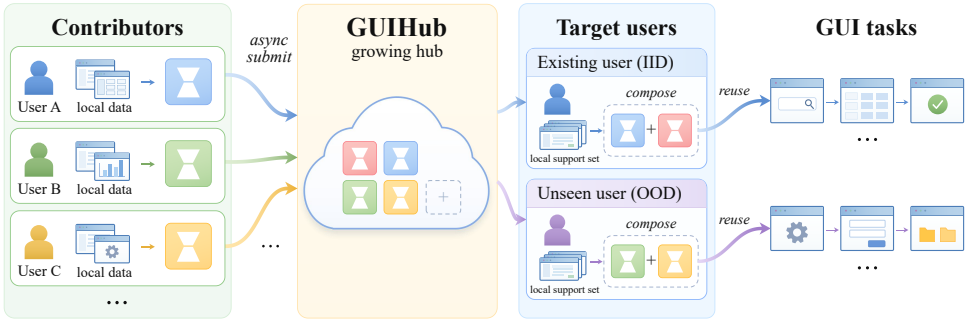


Figure 1: Overview of GUIHub. Contributors train adapters on local data and submit them to a growing Hub. Given a small local support set, GUIHub selects and composes adapters for existing (IID) and unseen (OOD) users, reusing the resulting composition across subsequent GUI tasks.

GUIHub evaluates adapters on a fixed global support set (GSS) and projects their response profiles into a shared capability space. A target's local support set specifies demand in this space, guiding adapter selection and composition weights. The selected LoRA branches are combined in output space with fixed weights during serving. New adapters are profiled on the same GSS and mapped into the existing capability space, allowing reuse without retraining existing adapters.

We evaluate GUIHub through offline action prediction on mobile GUI trajectories from FedMABench, examining personalization for in-hub clients, generalization to held-out clients, mixed-task composition, and Hub expansion. Across three backbones spanning different model families and scales, GUIHub achieves the highest IID and OOD Step Success Rate, outperforming the strongest baseline by 0.90–5.61 percentage points (pp). Expanding the Hub from 8 to 32 adapters improves pooled Step Success Rate by 6.16–7.61 pp across all three backbones on the same 32 target clients. With prebuilt Hub assets, GUIHub also achieves the lowest measured setup-plus-execution time on the 160-episode Qwen3.5-4B workload, yielding a 2.08–$5.61\times$ speedup over the evaluated baselines.

Our contributions are:

1. **An evolving Hub for GUI agents.** We introduce GUIHub for asynchronous sharing of independently trained GUI adapters, using a target support set to select and compose adapters for personalized reuse.
2. **Behavior-grounded selection and composition.** We connect target demand with measured adapter capabilities through shared GUI tasks. This enables adapter selection and weighting from an unlabeled local support set, producing a composition reused across subsequent requests.
3. **Four capabilities of GUIHub.** Across three backbones, we evaluate personalization, generalization, mixed-task composition, and Hub growth. GUIHub achieves the highest IID/OOD Step SR among the evaluated methods and improves overall as the Hub expands; a separate timing study shows lower setup-plus-execution cost.

2 Related Work

Adapter reuse and composition. LoRA enables efficient task specialization by learning low-rank updates to a shared frozen backbone (Hu et al., 2022). Adapter-reuse methods vary in selection and weighting. LoRAHub optimizes task-level composition weights using a few labeled target examples (Huang et al., 2024), whereas LoraRetriever retrieves and composes adapters based on the current input's semantic representation (Zhao et al., 2024). Wang et al. (2024) optimize instance-specific LoRA compositions using labeled examples, then retrieve one semantically per query. LoGo uses adapter activations from a probe forward pass for training-free, instance-level selection and weighting (Lee et al., 2026). LoRAuter uses validation performance and task embeddings to select adapters and weight their output contributions for each query (Dhasade et al., 2026). GUIHub instead combines shared behavioral profiles with an unlabeled local support set to determine a composition reused across a target workload.

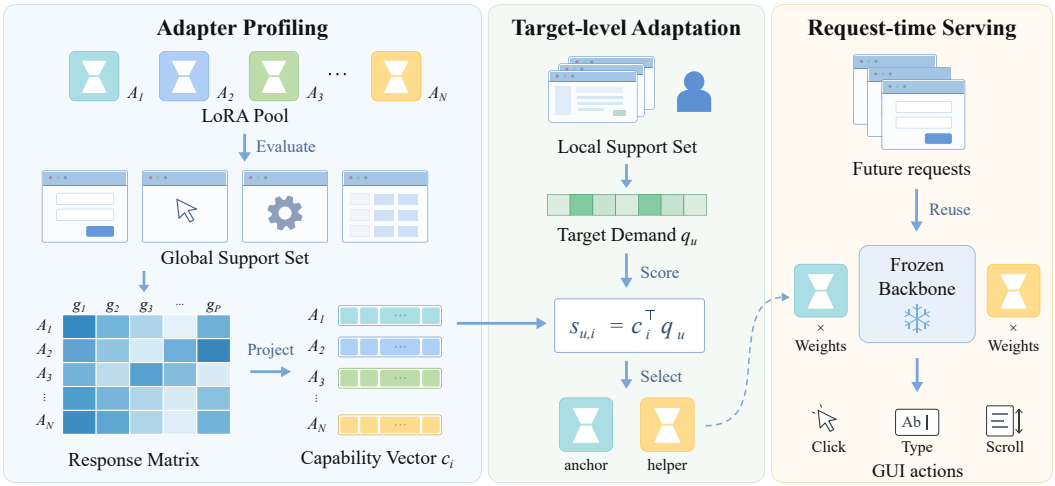


Figure 2: GUIHub workflow. **Adapter profiling:** responses on a shared global support set define adapter capability vectors. **Target-level adaptation:** a local support set specifies target demand, which guides adapter selection and weighting; an anchor–helper route is shown. **Request-time serving:** the resulting composition is reused with the frozen backbone across subsequent requests.

Personalized and decentralized GUI agents. FedMABench and FedGUI study GUI agents under decentralized data heterogeneity, including variation across users, platforms, devices, and operating systems (Wang et al., 2025; 2026). Related federated adaptation methods improve learning from distributed data through parameter-efficient updates (Ye et al., 2024; Sun et al., 2024), while asynchronous aggregation relaxes synchronized client participation (Nguyen et al., 2022). HiLoRA learns global, cluster-level, and client-specific adapters through hierarchical federated optimization for personalized vision models (Peng et al., 2026). These approaches organize collaborative training; GUIHub focuses on reusing independently completed adapters after training, with personalization defined by differences in target task distributions.

3 Methodology

3.1 Overview

GUIHub organizes independently trained GUI LoRA adapters through their behavior on a shared set of GUI episodes. Given a small LSS from a target client, it first infers the target's demand over these episodes, then matches this demand against the adapters' behavioral capabilities, and finally constructs a reusable target-level composition. Figure 2 illustrates the workflow.

We consider a frozen multimodal backbone f_0 and a pool of interface-compatible LoRA adapters $\mathcal{H} = \{A_i\}_{i=1}^{N}$. A shared global support set (GSS), $\mathcal{G} = \{g_j\}_{j=1}^{P}$, contains GUI episodes with reference actions for adapter profiling. A target client u provides a representative local support set (LSS), $\mathcal{L}_u$, whose instructions and initial screenshots describe its task distribution; routing uses no local action labels.

3.2 Adapter Profiling

3.2.1 Behavioral Profiles

We compare adapters on the fixed GSS using teacher-forced history. A soft behavioral score retains both action-type correctness and graded parameter agreement. Let z_{ijt} indicate that adapter A_i's prediction at step t of episode g_j parses and has the correct action type, and let $p_{ijt} \in [0, 1]$ measure agreement with the reference parameters. The response matrix $R \in \mathbb{R}^{N \times P}$ averages these scores

within each episode:

$$\sigma_{ijt} = z_{ijt}\left[\tfrac{1}{2} + \tfrac{1}{2}p_{ijt}\right], \qquad R_{ij} = \frac{1}{|\mathcal{V}_j|}\sum_{t\in\mathcal{V}_j}\sigma_{ijt}. \tag{1}$$

Here $\mathcal{V}_j$ contains the episode's valid non-terminal steps. The GSS balances task coverage and adapter discrimination; Appendix A.2 gives its selection objective and Appendix A.5 defines the parameter scores.

3.2.2 Capability Representation

Let $\mu \in \mathbb{R}^P$ denote the mean GSS response of a fixed set of reference adapters and $B \in \mathbb{R}^{P\times K}$ a shared projection basis. We represent adapter A_i by its capability vector

$$c_i = B^\top \left(R_{i,:}^\top - \mu\right).$$

We obtain B from the top K right singular vectors of the column-centered reference responses and freeze B and μ. New adapters are profiled on the same GSS and represented using this mapping.

3.3 User–Adapter Matching

We estimate which GSS episodes are relevant to the target by comparing their task semantics with the LSS. A frozen multimodal encoder maps each episode's instruction and initial screenshot to an embedding. Let $e_{u,\ell}$ denote the embedding of the ℓ-th support episode and e_j^{G} the embedding of GSS episode g_j. For each support episode, let $\mathcal{N}_M(u,\ell)$ be the indices of the M GSS episodes with the largest cosine similarity to $e_{u,\ell}$; all other episodes receive zero relevance. We then compute

$$\rho_{u,\ell}(j) = \mathrm{Softmax}_{j\in\mathcal{N}_M(u,\ell)}\left(\frac{\cos(e_{u,\ell}, e_j^{\mathrm{G}})}{\tau_{\mathrm{rel}}}\right).$$

Here, $\tau_{\mathrm{rel}} > 0$ is the relevance temperature. We average the support-episode distributions to obtain the target-specific relevance vector

$$\rho_u = \frac{1}{|\mathcal{L}_u|}\sum_{\ell=1}^{|\mathcal{L}_u|}\rho_{u,\ell}.$$

The target demand and adapter compatibility score are then

$$q_u = B^\top \rho_u, \qquad s_{u,i} = c_i^\top q_u.$$

Semantic relevance identifies which shared tasks matter to the target, while measured behavior determines adapter compatibility.

3.4 Routing and Composition

GUIHub uses an anchor-aware two-adapter route: when available, a source-associated adapter serves as the anchor, and an additional adapter is selected as the helper. For an IID target, the anchor is its own adapter if present in the Hub. For a cross-client mixture, it is the highest-scoring available IID adapter among its known source clients. The helper is the highest-scoring remaining candidate. If no anchor is available, we select the two highest-scoring candidates. Let S_u denote the indices of the selected adapters. We then compute composition weights only over the selected set:

$$\pi_{u,i} = \frac{\exp(s_{u,i}/\tau_{\mathrm{route}})}{\sum_{j\in S_u}\exp(s_{u,j}/\tau_{\mathrm{route}})}, \qquad i \in S_u.$$

Here, $\tau_{\mathrm{route}} > 0$ is the routing temperature.

At each LoRA-augmented linear layer k with frozen backbone weight $W_{0,k}$ and input h_k, we sum the selected adapters' weighted output contributions:

$$h'_k = W_{0,k}h_k + \gamma \sum_{i \in S_u} \pi_{u,i} \Delta W_{i,k} h_k, \qquad \Delta W_{i,k} = \frac{\alpha_i}{r_i} U_{i,k} V_{i,k}.$$

Here, $U_{i,k}$ and $V_{i,k}$ are the LoRA factors of adapter A_i at layer k, r_i and α_i are its rank and scaling hyperparameter, and γ is the fixed composition scale.

The selected adapters and their weights are determined once during target adaptation and reused for subsequent requests, without per-action retrieval.

4 Experiments

We evaluate personalization, held-out transfer, mixed-client composition, and Hub expansion, followed by efficiency measurements, component ablations, and routing diagnostics.

4.1 Evaluation Framework

Benchmark and target clients. We use the AndroidControl mobile GUI trajectories (Li et al., 2024) provided through FedMABench (Wang et al., 2025). Each episode contains an instruction, screenshots, and an action sequence. Using balanced spherical clustering of training-side representations, we construct 32 semantic clients with distinct task distributions: 24 in-hub clients (IID) and 8 held-out clients (OOD). These are evaluation units rather than real user identities. We use fixed client assignments throughout training and evaluation; Appendix A.1 describes their construction and training-instruction task profiles.

Data partitions and support sets. Each client has 300 training episodes, 50 GSS candidates, 20 LSS candidates, 20 development episodes, and 50 test episodes. The GSS contains 32 episodes from each in-hub client, totaling 768 episodes selected for task coverage and adapter discrimination. For each single-client target, we select 10 LSS episodes by semantic coverage using instruction and initial-screenshot embeddings, with the separate development episodes serving as the coverage reference. Selection uses no local action labels or test outcomes. Mixed targets combine source-specific support subsets as described in Section 4.4. The IID and OOD test sets contain 1,200 and 400 episodes, respectively. Appendix A.2 details the selection and separation rules.

Backbones and adapter training. We build separate adapter pools for Qwen3.5-4B, Qwen3.5-9B (Qwen Team, 2026), and Gemma-4-E4B-it (Gemma Team, 2026). Each client independently trains one rank-16 LoRA adapter with $\alpha = 32$ for three epochs, keeping the backbone, vision encoder, and multimodal aligner frozen. Selection and composition operate within one backbone. Training settings and the final-checkpoint rule appear in Appendix A.3.

Baselines and information access. We compare with the backbone without adapters (Base), LoRAHub (Huang et al., 2024), LoGo (Lee et al., 2026), and LoraRetriever (Zhao et al., 2024). The adapter-based methods draw from the same available Hub; all methods use the same test episodes and action evaluator within each comparison. LoRAHub searches composition weights using five labeled local episodes. GUIHub uses LSS instruction and initial-screenshot embeddings together with Hub-side behavioral profiles, without reading local action labels. Both produce a target-level composition. LoraRetriever retrieves adapters from the current episode query, whereas LoGo routes using action-step forward signals. All methods observe the current instruction and screenshot when predicting actions. Appendix A.4 specifies their GUI adaptations and information access.

Evaluation and metrics. We evaluate offline action prediction using greedy decoding and teacher-forced history: each step receives the preceding ground-truth actions. We report Type Accuracy (Type), Ground Accuracy (Ground), and Step Success Rate (Step SR), with Step SR as the primary metric. Type and Step SR are evaluated over valid non-terminal steps, while Ground is evaluated only on reference tap and long-press steps. Step SR requires both the correct action type and matching executable parameters. For each metric, we sum correct predictions and eligible steps over the evaluated episodes and report their ratio as a percentage. Appendix A.5 specifies the action-matching rules and aggregation details.

Table 1: Action prediction performance (%) on 24 in-hub (IID) and 8 held-out (OOD) clients. Bold indicates the best result in each column within each backbone.

Backbone	Method	IID			OOD		
		Type ↑	Ground ↑	Step SR ↑	Type ↑	Ground ↑	Step SR ↑
Qwen3.5-4B	Base	44.16	10.63	22.60	39.28	6.55	26.58
	LoRAHub	79.56	35.74	45.69	79.51	26.29	47.08
	LoGo	77.63	60.05	60.54	78.41	58.52	63.62
	LoraRetriever	79.18	61.07	62.24	78.93	59.74	64.43
	GUIHub	**81.71**	**63.08**	**64.98**	**81.84**	**61.31**	**67.00**
Qwen3.5-9B	Base	53.69	16.39	26.57	46.41	8.56	27.72
	LoRAHub	79.33	56.00	58.98	77.18	49.17	57.87
	LoGo	79.38	67.36	64.49	77.79	64.37	64.57
	LoraRetriever	81.05	67.63	66.24	79.22	64.80	65.95
	GUIHub	**81.58**	**71.82**	**70.45**	**80.22**	**72.58**	**71.56**
Gemma-4-E4B-it	Base	61.20	13.11	14.02	49.93	9.08	12.93
	LoRAHub	76.56	40.20	42.19	75.70	39.56	45.51
	LoGo	76.40	39.02	40.17	74.94	35.28	41.56
	LoraRetriever	77.88	43.96	44.69	**75.89**	**42.10**	47.27
	GUIHub	**78.33**	**45.80**	**46.44**	**75.89**	42.01	**48.17**

4.2 Personalization for In-Hub Clients (RQ1)

Can GUIHub improve personalization for clients whose adapters are already available in the Hub?

Setup. We evaluate all methods on 1,200 test episodes from the 24 in-hub clients, with the 24 in-hub adapters forming the available Hub for each backbone. GUIHub uses each target's ten-episode LSS to select a helper alongside its own adapter as the anchor, then reuses the composition across that target's test episodes.

Results. Compared with LoraRetriever in Table 1, GUIHub improves IID Step SR by 2.74, 4.21, and 1.75 pp on Qwen3.5-4B, Qwen3.5-9B, and Gemma, respectively. Ground improves by 2.00, 4.19, and 1.84 pp. Paired-episode 95% intervals for all three Step SR gains exclude zero (Appendix D.5), supporting adaptation to in-hub task distributions from a small LSS.

GUIHub also outperforms directly loading each target's own LoRA on Qwen3.5-4B, improving IID Step SR by 18.08 pp (Appendix D.1, Table 12).

4.3 Generalization to Held-Out Clients (RQ2)

How well does GUIHub generalize to held-out clients without client-specific adapters in the Hub?

Setup. We evaluate the eight held-out clients on 400 test episodes using the same 24-adapter Hub as in RQ1. No adapter trained on a held-out client is available to any method. GUIHub constructs a fixed two-adapter composition from each target's ten-episode LSS using anchor-free selection.

Results. GUIHub achieves OOD Step SR of 67.00%, 71.56%, and 48.17% on Qwen3.5-4B, Qwen3.5-9B, and Gemma, respectively, exceeding LoraRetriever by 2.57, 5.61, and 0.90 pp (Table 1). GUIHub achieves the highest OOD Step SR across all three backbones, supporting effective transfer to held-out clients without dedicated adapters. Appendix D.5 reports paired-episode confidence intervals.

4.4 Composition for Mixed Client Distributions (RQ3)

Can GUIHub adapt to mixed task distributions spanning multiple clients with a single reusable adapter composition?

Setup. We simulate users whose workloads span multiple client distributions by constructing 16 mixed targets, each containing 60 complete test episodes drawn from its source clients. **Primary** comprises nine pairs spanning IID/IID, IID/OOD, and OOD/OOD sources at near, medium, and far demand distances; each contributes $5 + 5$ LSS and $30 + 30$ test episodes. **Extension** comprises

Table 2: Step SR (%) on Primary, Extension, and Sensitivity mixtures. Results are pooled within each mixture group. Bold indicates the best result within each backbone and group.

	Qwen3.5-4B			Qwen3.5-9B		
Method	Primary ↑	Extension ↑	Sensitivity ↑	Primary ↑	Extension ↑	Sensitivity ↑
Base	22.88	24.18	24.24	25.92	26.15	27.58
LoRAHub	44.83	49.12	44.75	57.03	64.61	59.16
LoGo	61.18	64.74	59.16	63.98	67.26	63.65
LoraRetriever	62.15	65.56	60.21	64.85	67.53	64.41
GUIHub	**64.71**	**69.29**	**62.98**	**68.58**	**72.76**	**70.42**

four balanced triples covering zero to three IID sources, with $3+3+3$ LSS and $20+20+20$ test episodes. **Sensitivity** reuses the three medium-distance Primary pairs with $4+6$ LSS and $24+36$ test episodes, testing a 40/60 mixture. Demand distance is one minus cosine similarity between frozen client demand vectors.

These LSS allocations apply to GUIHub; LoRAHub uses five labeled local episodes per target. All methods draw from the 24-adapter in-hub pool. GUIHub constructs one fixed composition per mixed target, using a source-associated adapter as the anchor when available and anchor-free selection otherwise. Source membership is known during target preparation, but per-request source labels are not provided at test time. LoraRetriever and LoGo retain episode- and step-level routing. We pool correct and eligible step counts within each mixture group, counting an episode separately when it occurs in different target workloads. Appendix B.1 gives source IDs, allocation rules, and prediction-reuse details.

Results. GUIHub exceeds LoraRetriever on Primary, Extension, and Sensitivity by 2.56, 3.74, and 2.77 pp on Qwen3.5-4B, and by 3.73, 5.23, and 6.01 pp on Qwen3.5-9B (Table 2). A fixed target-level composition can therefore serve these mixed task distributions competitively without repeating adapter selection for each request.

4.5 Evolving Capability with Hub Expansion (RQ4)

How does GUIHub's performance evolve as independently trained adapters are added to the Hub?

Setup. For each backbone, we order 32 independently trained adapters by ascending mean GSS response and expand the available Hub from 8 to 32 adapters in increments of four. At each stage, all methods draw from the same available Hub and are evaluated on the same 32 target clients and 1,600 test episodes. Target support sets and decoding settings remain fixed. GUIHub rebuilds its target compositions as the pool expands, using a target's own adapter as the anchor when it becomes available and anchor-free selection otherwise. The capability basis is fitted once to the 24 IID reference adapters before expansion and remains frozen; newly admitted adapters are profiled and projected into this space. This protocol evaluates serving-pool expansion under a pre-established representation. IID/OOD labels retain their original client assignments. Appendix B.3 gives arrival orders, split-wise curves, and results for the fixed initially covered clients.

Results. At Hub32, GUIHub reaches 66.04%, 70.75%, and 47.48% Step SR on Qwen3.5-4B, Qwen3.5-9B, and Gemma, respectively, exceeding LoraRetriever by 2.88, 4.78, and 1.33 pp (Figure 3). The paired-episode 95% intervals for these gains exclude zero (Appendix D.5). GUIHub leads at every evaluated Hub size except Hub8 on the two Qwen backbones. From Hub8 to Hub32, pooled Step SR across all 32 targets improves by 6.16, 7.61, and 6.41 pp, respectively. For the fixed eight targets per backbone whose own adapters are already present at Hub8, pooled Step SR also increases by 6.87, 2.66, and 0.55 pp, respectively (Table 9). The gains are therefore not solely attributable to the later arrival of targets' own adapters.

4.6 Efficiency Analysis

Setup. We evaluate Qwen3.5-4B with the 24-adapter pool from RQ4 on 32 targets, selecting five fixed test episodes per target (160 episodes and 998 action executions). Each method runs three repeats on NVIDIA A100-SXM4-80GB GPUs with BF16 and generation batch size one. Timing starts from

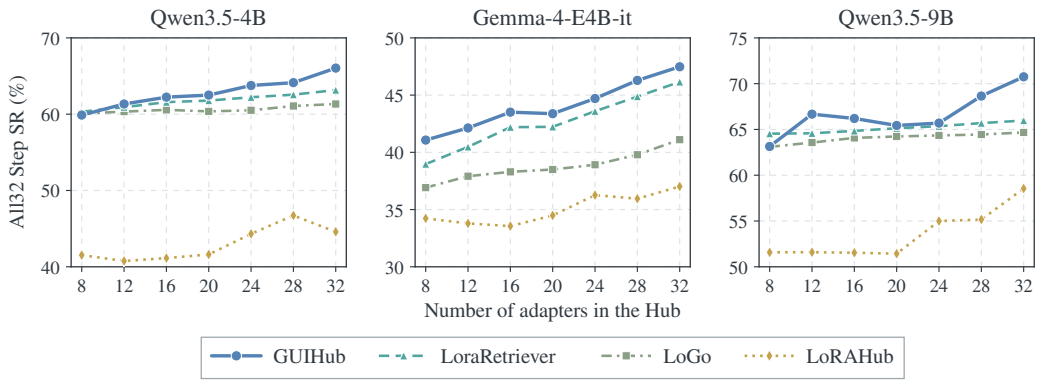


Figure 3: Step SR (%) as the Hub expands from 8 to 32 adapters, evaluated on the same 32 clients and 1,600 test episodes. Panels use backbone-specific vertical ranges.

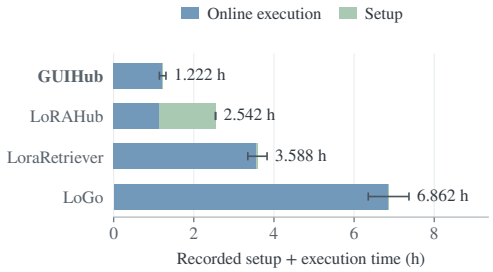


Figure 4: Recorded time for the 160-episode workload. Bars show mean accumulated time over three repeats, split into setup and online execution; error bars indicate the sample standard deviation of total time. Online execution includes retrieval/fusion for LoraRetriever and routing for LoGo. Timing boundaries are detailed in Appendix C.

prebuilt Hub-side assets. We report accumulated durations of the recorded setup and online execution intervals. Appendix C specifies timing boundaries, warmup, and hardware allocation.

Results. GUIHub records 1.222 hours for the fixed workload, compared with 2.542, 3.588, and 6.862 hours for LoRAHub, LoraRetriever, and LoGo: ratios of 2.08, 2.94, and 5.61 (Figure 4). Lower setup cost drives the advantage over LoRAHub, whose online latency is slightly lower; GUIHub has lower online execution time than LoraRetriever and LoGo. These measurements support lower recorded setup-plus-execution cost on this workload.

4.7 Ablation and Analysis

Component ablations. On Qwen3.5-4B, we replace behavioral routing scores with semantic relevance scores while keeping the candidate pool, two-adapter selection, anchor rule, weighting rule, composition scale, decoder, and test episodes fixed. A second variant removes the anchor and selects the two highest-scoring adapters. We pool counts over the IID, OOD, or all 16 mixed-client test sets. The no-anchor experiment covers IID and mixed clients; its OOD result is not independently measured.

Table 3: Qwen3.5-4B component ablations: pooled Step SR (%). † denotes the OOD no-anchor result reused from RQ2; bold marks column maxima.

Variant	IID	OOD	Mixed (All)
GUIHub	**64.98**	**67.00**	**65.62**
Semantic routing	63.96	61.82	61.76
Without anchor	63.99	**67.00**†	65.16

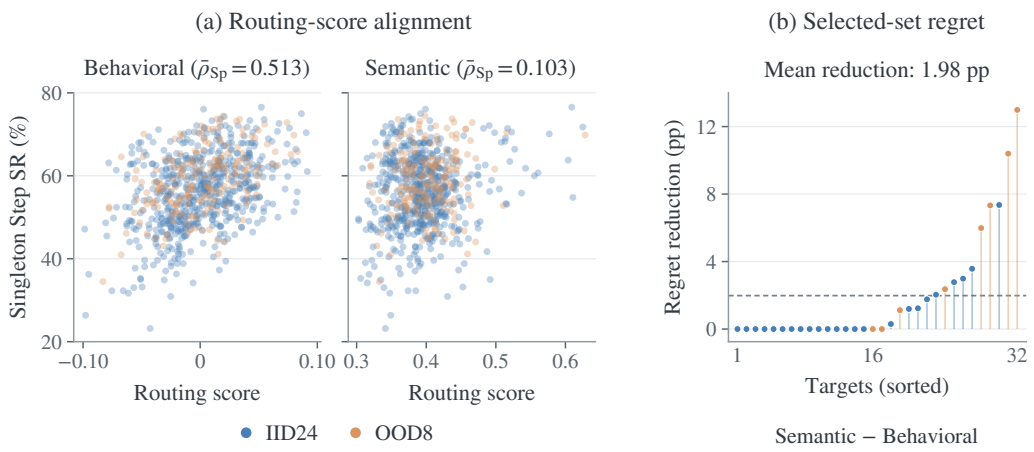


Figure 5: Routing diagnostics on Qwen3.5-4B. (a) Routing scores versus singleton Step SR for 768 target–adapter pairs; $\bar{\rho}_{Sp}$ is the mean within-target Spearman correlation. (b) Per-target regret reduction (semantic minus behavioral), sorted across 32 targets; the dashed line marks the mean. Colors denote IID/OOD targets.

Behavioral routing consistently outperforms matched semantic routing, improving Step SR by 1.02, 5.18, and 3.86 pp on IID, OOD, and mixed clients, respectively (Table 3); the IID and OOD gains also have paired-episode 95% intervals excluding zero (Appendix D.5).

Without anchors, GUIHub still exceeds LoraRetriever by 1.75 pp on IID and 2.46 pp on mixed clients, while anchor-aware routing provides modest additional gains of 0.99 and 0.46 pp (Table 3).

Removing the low-rank routing projection has little effect on Step SR (Appendix D.2). Across the tested adapter counts, support budgets, capability dimensions, and relevance budgets, Step SR changes by less than 1 pp (Appendix D.3).

Routing diagnostics. We independently evaluate all 24 candidate adapters on each target's 50 test episodes using Qwen3.5-4B. Each singleton uses weight one and scale $\gamma = 1.5$, with the same decoder and teacher-forced history. We compute a Spearman correlation between scores and singleton Step SR within each target and then average across targets. *Selected-set regret* is the gap between the best singleton in the candidate pool and the best singleton among the two selected adapters, averaged across targets in pp. This diagnostic uses frozen selected sets and does not apply composition weights.

Behavioral scores rank singleton capability more consistently than semantic scores: the mean per-target Spearman correlations across all 32 clients are 0.513 and 0.103, respectively (Figure 5a). Behavioral selected sets reduce mean regret from 3.07 to 1.09 pp, an average reduction of 1.98 pp (Figure 5b). Together, these diagnostics support behavioral profiles for adapter selection. Appendix D.4 reports split-wise statistics and uncertainty intervals. These singleton diagnostics do not evaluate composition optimality.

5 Conclusion

We presented GUIHub, an asynchronous framework for sharing and composing independently trained GUI LoRA adapters. Its core idea is to use semantic relevance from a small local support set to weight behavioral evidence on shared GUI tasks, matching adapter capabilities to user demand without local action labels. The resulting target-level composition is reused across requests, while new adapters join the shared capability space without retraining existing adapters. Across three backbones, offline mobile action prediction experiments show higher Step Success Rate than the evaluated baselines on in-hub and held-out clients, together with overall gains as the Hub expands. A separate Qwen3.5-4B timing study also shows lower recorded setup-plus-execution time with prebuilt Hub assets, supporting behavior-grounded adapter reuse in a growing shared Hub.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on research methodology or experiments, assist with translation, support qualitative and thematic data analysis, and interpret results. We have not used generative AI tools to assist in writing proofs, implement methods, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, or clean and reformat datasets. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 specifies adapter profiling, capability representation, target matching, and composition. Appendix A documents the client partitions, support-set selection, training and inference settings, prompt structure, baseline adaptations, and action evaluation rules. Appendices B.1 and B.3 describe the mixed-client construction and Hub-expansion protocols, including the sampling rules and adapter arrival orders. Appendix C reports the hardware, execution conditions, and timing boundaries of the efficiency measurements. Appendices D.2 and D.3 detail the component ablations and hyperparameter sensitivity experiments, respectively. Appendix D.5 specifies the paired-episode bootstrap procedure and the scope of its uncertainty estimates.

REFERENCES

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, 2024. doi: 10.18653/v1/2024.acl-long.505. URL https://doi.org/10.18653/v1/2024.acl-long.505.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114, 2023. doi: 10.52202/075280-1220. URL https://doi.org/10.52202/075280-1220.

Akash Dhasade, Anne-Marie Kermarrec, Igor Pavlovic, Diana Petrescu, Rafael Pires, Mathis Randl, and Martijn de Vos. Effective LoRA adapter routing using task representations. arXiv preprint arXiv:2601.21795, 2026. URL https://arxiv.org/abs/2601.21795.

Gemma Team. Gemma 4 technical report. arXiv preprint arXiv:2607.02770, 2026. URL https://arxiv.org/abs/2607.02770.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. CogAgent: A visual language model for GUI agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14281–14290, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Hong_CogAgent_A_Visual_Language_Model_for_GUI_Agents_CVPR_2024_paper.html.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. LoraHub: Efficient cross-task generalization via dynamic LoRA composition. In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2307.13269.

Seungeon Lee, Soumi Das, Manish Gupta, and Krishna P. Gummadi. LoRA on the go: Instance-level dynamic LoRA selection and merging. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 39583–39601, 2026. doi: 10.18653/v1/2026.acl-long.1837. URL https://doi.org/10.18653/v1/2026.acl-long.1837.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Qwen3-VL-Embedding and Qwen3-VL-Reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. arXiv preprint arXiv:2601.04720, 2026. URL https://arxiv.org/abs/2601.04720.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on UI control agents. In *Advances in Neural Information Processing Systems*, volume 37, pp. 92130–92154, 2024. doi: 10.52202/079017-2925. URL https://doi.org/10.52202/079017-2925.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. ShowUI: One vision-language-action model for GUI visual agent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19498–19508, 2025. URL https://arxiv.org/abs/2411.17465.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 3581–3607, 2022. URL https://proceedings.mlr.press/v151/nguyen22b.html.

Hongyi Nie, Xunyuan Liu, Yudong Bai, Yaqing Wang, Yang Liu, Quanming Yao, and Zhen Wang. PSPA-Bench: A personalized benchmark for smartphone GUI agent. arXiv preprint arXiv:2603.29318, 2026. URL https://arxiv.org/abs/2603.29318.

Zihao Peng, Nan Zou, Jiandian Zeng, Guo Li, Ke Chen, Boyuan Li, and Tian Wang. HiLoRA: Hierarchical low-rank adaptation for personalized federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 31746–31757, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Peng_HiLoRA_Hierarchical_Low-Rank_Adaptation_for_Personalized_Federated_Learning_CVPR_2026_paper.html.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. UI-TARS: Pioneering automated GUI interaction with native agents. arXiv preprint arXiv:2501.12326, 2025. URL https://arxiv.org/abs/2501.12326.

Qwen Team. Qwen3.5: Towards native multimodal agents. Qwen blog, 2026. URL https://qwen.ai/blog?id=qwen3.5.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, pp. 406–441, 2025. URL https://openreview.net/forum?id=il5yUQsrjC.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. AndroidInTheWild: A large-scale dataset for Android device control. In *Advances in Neural Information Processing Systems*, volume 36, pp. 59708–59728, 2023. doi: 10.52202/075280-2609. URL https://doi.org/10.52202/075280-2609.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving LoRA in privacy-preserving federated learning. In *International Conference on Learning Representations*, pp. 17978–17994, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4e243e95c913b367775d71d7182b99d9-Abstract-Conference.html.

WenHao Wang, Zijie Yu, Rui Ye, Jianqing Zhang, Guangyi Liu, Liang Liu, Siheng Chen, and Yanfeng Wang. FedMABench: Benchmarking mobile GUI agents on decentralized heterogeneous user data. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 26387–26408, 2025. doi: 10.18653/v1/2025.emnlp-main.1341. URL https://doi.org/10.18653/v1/2025.emnlp-main.1341.

WenHao Wang, Haoting Shi, Mengying Yuan, Yiquan Lin, Panrong Tong, Hanzhang Zhou, Guangyi Liu, Pengxiang Zhao, Yue Wang, and Siheng Chen. FedGUI: Benchmarking federated GUI agents across heterogeneous platforms, devices, and operating systems. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 28747–28767, 2026. doi: 10.18653/v1/2026.findings-acl.1436. URL https://doi.org/10.18653/v1/2026.findings-acl.1436.

Zhiqi Wang, Shizhu He, Kang Liu, and Jun Zhao. Instance-level dynamic LoRAs composition for cross-task generalization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5699–5708, 2024. doi: 10.18653/v1/2024.findings-emnlp.326. URL https://doi.org/10.18653/v1/2024.findings-emnlp.326.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. OS-ATLAS: A foundation action model for generalist GUI agents. In *International Conference on Learning Representations*, pp. 5090–5108, 2025. URL https://openreview.net/forum?id=n9PDaFNi8t.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094, 2024. doi: 10.52202/079017-1650. URL https://doi.org/10.52202/079017-1650.

Rui Ye, Wenhao Wang, Jingyi Chai, Dihan Li, Zexi Li, Yinda Xu, Yaxin Du, Yanfeng Wang, and Siheng Chen. OpenFedLLM: Training large language models on decentralized private data via federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6137–6147, 2024. doi: 10.1145/3637528.3671582. URL https://doi.org/10.1145/3637528.3671582.

Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. UFO: A UI-focused agent for Windows OS interaction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 597–622, 2025a. doi: 10.18653/v1/2025.naacl-long.26. URL https://doi.org/10.18653/v1/2025.naacl-long.26.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. AppAgent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–20, 2025b. doi: 10.1145/3706598.3713600. URL https://doi.org/10.1145/3706598.3713600.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. LoraRetriever: Input-aware LoRA retrieval and composition for mixed tasks in the wild. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4447–4462, 2024. doi: 10.18653/v1/2024.findings-acl.263. URL https://doi.org/10.18653/v1/2024.findings-acl.263.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Table 4: Episode pools before support selection. Counts refer to complete episodes. The selected GSS and LSS are subsets of their respective candidate pools.

Partition	Per client	IID (24 clients)	OOD (8 clients)
Training	300	7,200	2,400
GSS candidates	50	1,200	400
LSS candidates	20	480	160
Development	20	480	160
Test	50	1,200	400

A Data and Evaluation Details

A.1 Semantic clients and task distributions

Our evaluation uses mobile GUI episodes from the AndroidControl-based release (Li et al., 2024) used with FedMABench (Wang et al., 2025). Each episode contains an instruction, screenshots, and an action sequence. We construct 32 semantic clients through balanced spherical clustering of training-side representations and assign the remaining partitions using the frozen centers. Clients represent heterogeneous task distributions rather than real user identities or mutually exclusive applications.

The 24 in-hub client IDs, in candidate order, are 11, 25, 24, 23, 0, 1, 2, 3, 17, 14, 16, 4, 5, 6, 18, 9, 12, 10, 13, 20, 7, 8, 19, and 15; the eight held-out IDs are 21, 22, 26, 27, 28, 29, 30, and 31. Each client contributes 300 training episodes, 50 GSS candidates, 20 LSS candidates, 20 development episodes, and 50 test episodes. Client assignments and test membership remain fixed throughout evaluation. Support selection and configuration use no test outcomes. IID/OOD denote these experimental roles and do not imply identical or disjoint task semantics.

Partition roles. Table 4 separates the available episode pools from the selected support sets. Adapter training uses the training partition. GSS selection uses the 1,200 candidates from the 24 IID clients, yielding 768 shared probes. The 400 candidates belonging to OOD clients do not enter this shared GSS. Local candidates and development episodes provide instruction/screenshot inputs for LSS coverage selection; their action labels are not used by that selection objective. Test episodes supply the held-out predictions and reference actions used to calculate accuracy. The existence of training data for an OOD client does not make its adapter available in RQ1–RQ3: the available Hub in those experiments remains the fixed pool of 24 in-hub adapters. RQ4 explicitly changes adapter availability as the Hub expands.

Task-distribution profiles. We characterize the 9,600 training instructions using 18 multi-label keyword indicators and 16 TF–IDF/NMF topics (Figure 6), without reading test inputs or images. Keyword indicators describe overlapping instruction patterns, while topic weights describe lexical mixtures; neither is an annotated task taxonomy. In total, 24.66% of instructions match no keyword indicator and three have no topic assignment. Application names remain among the lexical features.

Across the 496 client pairs, square-root Jensen–Shannon distances between topic mixtures have mean 0.6682 and median 0.6954 (Figure 7). With a fixed topic basis, mean split-half within-client and between-client distances are 0.0853 and 0.6708, compared with 0.0941 for a shuffled reference. The split halves contain 150 episodes and shuffled groups contain 300, so this comparison is descriptive rather than a matched significance test. These profiles illustrate heterogeneity in the constructed client workloads.

A.2 Support-set selection

GSS discrimination filtering. The GSS contains 768 episodes, with 32 selected from each in-hub client. We first evaluate all 1,200 candidates (50 per client) with the 24 Qwen3.5-4B reference adapters using the soft episode score defined in Appendix A.5. Let $R^{\text{cand}} \in \mathbb{R}^{24\times 1200}$ be this

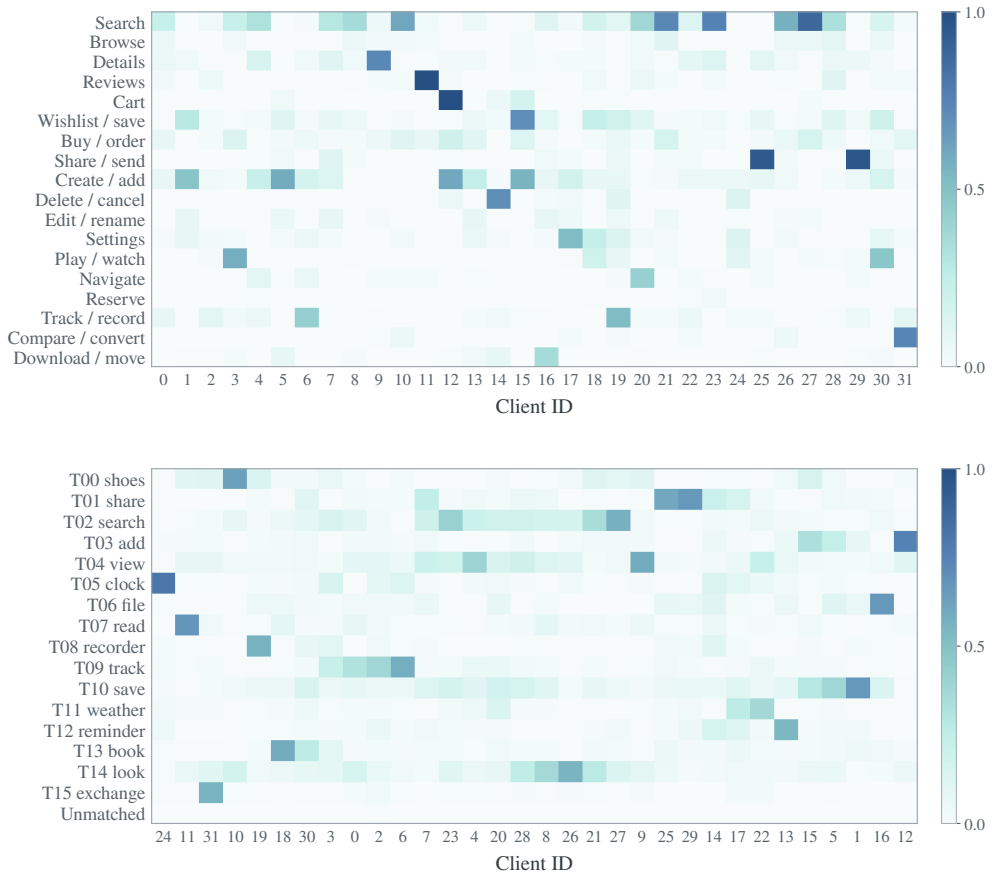


Figure 6: Training-instruction profiles for 32 clients, each with 300 episodes. Top: fractions matching 18 overlapping keyword indicators. Bottom: mean normalized weights of 16 lexical topics and an unmatched category, ordered by hierarchical clustering of topic distances. Topic labels show the topic ID and highest-weight term. Both color scales span 0–1.

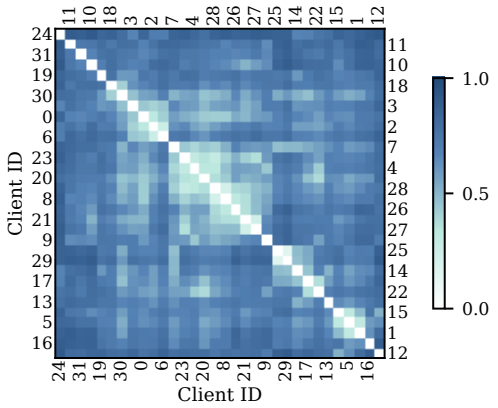


Figure 7: Square-root Jensen–Shannon distances between client topic mixtures, estimated from training instructions. Numeric client IDs alternate between opposite axes for readability; rows and columns share the same clustering order. Zero indicates identical distributions and one disjoint distributions.

candidate response matrix. Candidate j receives discrimination score

$$v_j = \frac{1}{24}\sum_{i=1}^{24}\left(R_{ij}^{\text{cand}} - \bar{R}_j^{\text{cand}}\right)^2, \qquad \bar{R}_j^{\text{cand}} = \frac{1}{24}\sum_{i=1}^{24} R_{ij}^{\text{cand}}. \tag{2}$$

Within each client, we remove the ten lowest-variance candidates, resolving ties by episode ID. This leaves a set $\mathcal{C}_c$ of 40 candidates for client c.

GSS coverage selection. Let e_j be the unit-normalized frozen multimodal embedding of candidate j's instruction and initial screenshot. We column-center R^{cand} and compute its rank-eight truncated SVD. If $V_8 \in \mathbb{R}^{1200\times 8}$ contains its right singular vectors, b_j is the unit-normalized jth row of V_8 (zero if its norm is at most 10^{-12}). Semantic and behavioral evidence contribute equally:

$$\kappa(j,k) = \tfrac{1}{2}e_j^\top e_k + \tfrac{1}{2}b_j^\top b_k, \qquad F_c(S) = \sum_{j\in\mathcal{C}_c}\max_{k\in S}\kappa(j,k). \tag{3}$$

We select $|S_c| = 32$ to maximize coverage approximately by greedy marginal gains, independently within each client. Starting with $S_c = \emptyset$ and coverage $h_j = -1$, each iteration selects

$$k^* = \arg\max_{k\in\mathcal{C}_c\setminus S_c}\sum_{j\in\mathcal{C}_c}\left[\max\{h_j,\kappa(j,k)\} - h_j\right], \qquad h_j \leftarrow \max\{h_j,\kappa(j,k^*)\}. \tag{4}$$

Episode ID resolves tied gains. The union of the 24 subsets is the fixed GSS reused across backbones; the 32-per-client budget preserves client coverage while the similarity objective reduces redundant probes. For each backbone, its own 24 IID reference adapters are profiled on this GSS to fit the final capability mean μ and basis B, which remain frozen during Hub expansion.

LSS coverage selection. For target u, let $\mathcal{C}_u$ contain its 20 local candidates and $\mathcal{D}_u$ its separate 20 development episodes. Using the same frozen instruction/screenshot encoder, greedy selection approximates

$$\max_{S\subseteq\mathcal{C}_u,\,|S|=10}\sum_{x\in\mathcal{D}_u}\max_{s\in S}\cos(e_x,e_s). \tag{5}$$

Episode ID resolves ties and fixes the selection order. This objective uses no local action labels or test outcomes. Mixed targets take source-specific prefixes of these orders, with counts given in Table 6. The reduced five-episode condition uses prefixes of $3+2$, $2+2+1$, and $2+3$ for Primary, Extension, and Sensitivity, respectively.

Semantic inputs and local data budget. Semantic comparisons use frozen 2,048-dimensional Qwen3-VL-Embedding-2B representations (Li et al., 2026) of the episode instruction and initial screenshot, with ℓ_2 normalization for cosine similarity. The ten LSS episodes are selected from 20 unlabeled candidates using 20 separate development inputs as the coverage reference, for a total of 40 distinct local inputs during support preparation. Thus the routing budget and the data accessed during support preparation are distinct. After selection, the target relevance vector is computed from the selected LSS alone, using the shared GSS embeddings as references.

A.3 Training and inference settings

Adapters are trained independently on Qwen3.5-4B, Qwen3.5-9B, or Gemma-4-E4B-it with LoRA rank 16, $\alpha = 32$, dropout 0.05, learning rate 5×10^{-5}, three epochs, batch size one, and eight gradient-accumulation steps. Training uses BF16, weight decay 0.1, a cosine learning-rate schedule, warmup ratio 0.03, maximum sequence length 4,096, maximum image size 100,352 pixels, gradient checkpointing, and seed 42. Loss is applied only to assistant action tokens. The backbone, vision encoder, and aligner remain frozen. We save epoch checkpoints and use the final third-epoch adapter, without selecting it on test performance.

The optimizer is fused AdamW with $(\beta_1,\beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and maximum gradient norm 0.5. Each client adapter is trained by an independent single-GPU job, giving an effective batch size of eight step examples. Sequence packing is disabled; examples exceeding the maximum token length are dropped by the training truncation policy. A training example contains the task instruction, the current screenshot, up to three preceding reference actions as text, and the next reference action as the assistant response.

LoRA is applied to language-model linear layers. In Qwen, these comprise attention query, key, value, and output projections (`q_proj`, `k_proj`, `v_proj`, `o_proj`); gated-delta input and output projections (`in_proj_qkv`, `in_proj_a`, `in_proj_b`, `in_proj_z`, `out_proj`); and feed-forward gate, up, and down projections. Gemma uses the attention and feed-forward projections, together with its per-layer input gate and associated projections (`per_layer_input_gate`, `per_layer_projection`, `per_layer_model_projection`). Parenthesized names identify modules in the corresponding model implementations. Vision and multimodal-alignment modules are excluded. Adapters are combined only within the same backbone.

GUIHub uses capability dimension $K = 8$, the 64 most relevant GSS episodes per support example, relevance temperature $\tau_{\text{rel}} = 0.07$, and routing temperature $\tau_{\text{route}} = 0.1$. The composition scale is $\gamma = 1.5$ for Qwen and $\gamma = 1.0$ for Gemma. Inference disables thinking, uses greedy decoding, temperature zero, top-p one, seed 42, batch size one, and a 128-token response budget. All accuracy experiments use preceding ground-truth actions as history.

Prompt and inference history. The shared policy instruction asks the model to predict exactly one next GUI action from the task goal, current screenshot, and preceding actions, and to return a compact JSON object without explanatory text. During evaluation, the first user message supplies the initial screenshot and episode instruction. Each subsequent user message supplies the next screenshot, states that the interface has been updated, and requests the next action. At step t, the input retains the screenshot turns through t and the reference assistant actions before t; the current reference action is withheld. This evaluation history differs from the single-screenshot training examples described above. Model predictions are scored but never inserted into later histories. Qwen receives the policy instruction as a system message, with reasoning output disabled; for Gemma, the policy text is prepended to the first user message. The action types and their evaluated parameters are described in Appendix A.5.

A.4 Baseline adaptation and information access

LoRAHub. LoRAHub (Huang et al., 2024) uses all available adapters when the pool contains at most 20; otherwise, it samples 20 without replacement from the ordered pool using Python's random sampler with seed 42. It likewise samples five complete labeled episodes from the target's ordered local support, with seed 42, and uses every action turn of those episodes. Nevergrad's derivative-free NGOpt optimizer searches signed coefficients initialized to zero for 40 objective evaluations, with bounds $[-1.5, 1.5]$. Let $\mathcal{T}_u$ be the supervised assistant-token positions across these turns. The objective is

$$\mathcal{J}_u(w) = -\frac{1}{|\mathcal{T}_u|} \sum_{a \in \mathcal{T}_u} \log p_w(y_a \mid x_a, y_{<a}) + \frac{0.05}{m} \sum_{i=1}^{m} |w_i|, \tag{6}$$

where m is the candidate count and x_a includes the corresponding teacher-forced GUI context. Prompt and padding tokens are masked, and the data loss is averaged over supervised tokens across all calibration turns. The coefficients are neither normalized nor pruned. At each adapted layer, fusion separately forms $\widetilde{U}_k = \sum_i w_i U_{i,k}$ and $\widetilde{V}_k = \sum_i w_i V_{i,k}$ in FP32, then uses the resulting LoRA factors with the trained scaling α/r. This factor fusion includes cross terms when the two factors are multiplied. The resulting composition is reused for target evaluation.

LoraRetriever. LoraRetriever (Zhao et al., 2024) makes one retrieval decision per episode from its instruction and initial screenshot. In our GUI adaptation, each IID adapter's semantic prototype is the mean embedding of the 32 fixed GSS episodes associated with its source client. Query and prototype vectors are normalized for cosine similarity, and the three highest-scoring adapters are selected; score ties preserve the candidate order. During Hub expansion, an admitted OOD adapter uses the first 32 eligible calibration episodes of its source client in lexicographic episode-ID order to form its prototype. Retrieval reads semantic inputs, without action labels or behavioral scores. The selected factors are averaged separately, $\widetilde{U}_k = \frac{1}{3} \sum_{i \in S} U_{i,k}$ and $\widetilde{V}_k = \frac{1}{3} \sum_{i \in S} V_{i,k}$, and their product is applied with the trained α/r scaling. The same fused adapter serves every step of that episode.

LoGo. LoGo (Lee et al., 2026) performs a routing forward pass before generating each action. It extracts the last non-padding prompt token's hidden state h_t at the input to the final language-model

Table 5: Action types and parameters used by the evaluator. Each prediction specifies one action type and its associated parameters.

Action	Evaluated parameters
Tap / long press	Screen coordinates and coordinate system
Type / answer / copy	Entered, returned, or copied text
Open application	Application name
Swipe	Scrolling effect or finger-motion direction
Key	Key name (e.g., back or home)
Wait	No parameter required for matching
Terminate	Completion status; scored separately from non-terminal actions

layer. For each candidate, it measures the norm of that layer's query-projection LoRA output:

$$a_{t,i} = \left\| \frac{\alpha_i}{r_i} U_{i,q} V_{i,q} h_t \right\|_2, \qquad S_t = \mathrm{TopK}_k(a_{t,:}),$$
$$k = \min(20, N), \qquad w_{t,i} = \frac{a_{t,i}}{\sum_{j \in S_t} a_{t,j}}. \tag{7}$$

These weights are shared across the adapted layers, which sum the selected LoRA outputs with their trained scaling. They are fixed during generation of that action and recomputed for the next action request. The first routing pass starts with coefficient one for every candidate; subsequent routing passes use the previously selected adapters and weights, including between episodes processed by the same loaded model. When an interrupted evaluation resumes, these adapters and weights are restored.

Information access. GUIHub uses selected LSS embeddings and labeled GSS behavioral profiles to construct a fixed target-level composition. LoRAHub instead optimizes on labeled local action sequences. LoraRetriever observes the current episode's semantic query, while LoGo observes action-step forward signals. GUIHub reads neither local action labels nor test queries when constructing its target route; all methods observe the current test inputs when predicting actions. These distinctions concern route construction, in addition to the common teacher-forced action history.

A.5 Evaluator and aggregation

Action format and parsing. The evaluator accepts a JSON object specifying one action and its parameters, either directly or embedded in generated text. Multiple embedded actions are treated as ambiguous and fail parsing. Action types require exact matches; synonymous names are not treated as equivalent. Table 5 lists the supported actions and evaluated parameters. Dataset actions are converted to this common format before evaluation: clicks are represented as taps, navigation as key actions, scrolling as swipes, and task completion as a terminal action. For scrolling, the content-movement direction and the opposite finger-motion direction are distinguished. Malformed reference actions cause evaluation to stop with an error rather than silently removing steps from the denominator.

Action accuracy. All methods use the same executable-action evaluator. Ground-truth terminal actions are recorded separately and excluded from Type and Step SR denominators. Parse failures count as incorrect and do not truncate subsequent teacher-forced steps. Type compares action types. Ground applies to reference tap and long-press actions; an incorrect type or failed parse counts as incorrect grounding. A point matches when it falls inside a reliable reference pixel bounding box; without one, its distance to the reference point, normalized by the screen diagonal, must be at most 0.14.

Step SR requires both the correct action type and matching parameters. Text requires normalized similarity of at least 0.5; swipes match the reference effect when provided and otherwise the gesture direction; key actions match normalized key names; wait requires the correct type. These action-based definitions follow the decomposition used in GUI evaluation (Wang et al., 2026) and differ from FedMABench's original text-matching metric.

Coordinate and parameter conventions. Coordinates are converted to pixels using the reference screen width W and height H: unit coordinates map to (Wx, Hy), and 0–1000 coordinates to

$(Wx/1000, Hy/1000)$. Explicitly normalized coordinate fields denote unit coordinates. If the coordinate system is omitted, a coordinate pair nested within the target object defaults to 0–1000, whereas a pair stored directly in the target object defaults to pixels. Missing or non-finite coordinates receive zero parameter credit; finite predicted coordinates are scored at their converted locations without clipping to screen boundaries. A reference bounding box is used only when its annotation marks it as reliable and both corners have finite coordinates within the screen; box boundaries count as inside. Otherwise the reference point determines the distance match. The text similarity below is used both as the soft parameter score and, at threshold 0.5, for Step SR; it also applies to app names. Key normalization strips surrounding whitespace, collapses internal whitespace, and ignores letter case, without treating different key names as synonyms. Swipe effects and directions require exact matches.

Soft behavioral score. For valid non-terminal step t, define $z_t = 1$ if the prediction parses and its action type matches the reference, and zero otherwise, suppressing the adapter and episode indices. The behavioral score and episode response are

$$\sigma_t = z_t \left[\lambda + (1-\lambda)p_t\right], \qquad \lambda = \tfrac{1}{2}, \qquad R_{ij} = \frac{1}{|\mathcal{V}_j|} \sum_{t \in \mathcal{V}_j} \sigma_{ijt}, \tag{8}$$

where $p_t \in [0,1]$ measures the parameter match and $\mathcal{V}_j$ contains the valid non-terminal steps of episode j. Incorrect types or failed parses receive zero; a correct type with an invalid parameter receives only the type credit, $1/2$. Parameter scores are defined as follows.

- **Tap/long-press:** $p_t = \max\{0, 1 - d_t/0.14\}$, where d_t is the screen-diagonal-normalized distance to the nearest point in a reliable reference bounding box, or to the reference point when no box is available. Thus any point inside the box receives $p_t = 1$.
- **Text:** normalize case and whitespace. Exact matches receive 1; one empty string receives 0. If one string contains the other, use the shorter-to-longer length ratio. Otherwise use set F1, $2|U \cap V|/(|U| + |V|)$, over whitespace tokens, or over non-space characters when either string contains a Chinese character; no overlap receives 0.
- **Swipe/key:** p_t is the binary match for the reference effect/direction or normalized key, respectively. **Wait:** $p_t = 1$ after the type check.

The continuous coordinate score is used for profiling; the Ground and Step SR metrics use the hard matching rules above.

Aggregation. For episode records $\mathcal{E}$, each reported accuracy is

$$100 \, \frac{\sum_{e \in \mathcal{E}} n_e^{\text{correct}}}{\sum_{e \in \mathcal{E}} n_e^{\text{applicable}}}. \tag{9}$$

Type and Step SR use valid non-terminal steps, while Ground uses grounding-applicable steps. Main results, ablations, and sensitivity all use this pooled aggregation. Mixed-client All includes all 16 mixture records, preserving repeated episodes when they belong to different target workloads. The IID, OOD, and combined test sets contain 6,796, 2,103, and 8,899 valid steps; Primary, Extension, and Sensitivity contain 2,893, 1,472, and 1,048, respectively. Differences are computed before rounding, and exact metric ties are highlighted together. Paired-episode confidence intervals are reported in Appendix D.5; repeated-training uncertainty is not estimated.

B Mixed-Client and Hub-Expansion Details

B.1 Mixture construction

We define three mixed-client groups, each with 60 test episodes per target. **Primary** contains nine balanced pairs covering IID/IID, IID/OOD, and OOD/OOD sources at near, medium, and far distances. **Extension** contains four balanced triples covering all combinations of IID/OOD source counts at medium distance. **Sensitivity** reuses the three medium-distance Primary pairs with a 40/60 test ratio; the OOD source is the majority for IID/OOD, and the second source listed in Table 6 is the majority otherwise. Table 6 gives all source IDs and support/test counts. **All** pools all 16 mixtures.

Table 6: Mixed-client construction grouped by Primary, Extension, and Sensitivity. Source IDs and per-source LSS/test counts share the same order. Distance is one minus cosine similarity between frozen client demand vectors, averaged over pairs for Extension.

Group	ID	Source clients	Distance	LSS	Test
Primary	P1	7/24	0.7071	5+5	30+30
	P2	4/10	0.4230	5+5	30+30
	P3	6/19	0.2213	5+5	30+30
	P4	14/27	0.6157	5+5	30+30
	P5	1/30	0.3207	5+5	30+30
	P6	5/29	0.1217	5+5	30+30
	P7	21/31	0.4530	5+5	30+30
	P8	26/31	0.2891	5+5	30+30
	P9	22/28	0.0920	5+5	30+30
Extension	E1	12/15/23	0.4537	3+3+3	20+20+20
	E2	20/25/26	0.3885	3+3+3	20+20+20
	E3	16/21/29	0.3254	3+3+3	20+20+20
	E4	28/29/30	0.2910	3+3+3	20+20+20
Sensitivity	S1	4/10	0.4230	4+6	24+36
	S2	1/30	0.3207	4+6	24+36
	S3	26/31	0.2891	4+6	24+36

Distance is one minus the cosine similarity between frozen client demand vectors, averaged over source pairs for Extension. Candidate source combinations are sorted by distance, with client-ID tuples breaking ties, separately within each IID/OOD composition. Near, medium, and far correspond to consecutive rank terciles: for n candidates, stratum $b \in \{0, 1, 2\}$ contains zero-based ranks $\lfloor nb/3 \rfloor$ through $\lfloor n(b+1)/3 \rfloor - 1$. Pairs are selected in IID/IID, IID/OOD, and OOD/OOD order, each from near to far; triples follow in order of three, two, one, and zero IID sources, using the middle tercile. If client c has already appeared n_c times, selection first minimizes $\sum_{c \in C}[(n_c+1)^2 - n_c^2]$ for candidate source set C, then its rank distance to the stratum midpoint, then the client-ID tuple. Exposure counts are updated after each selection.

For source client c, its 50 test episodes are sorted lexicographically by episode ID and shuffled with Python's `random.Random(42+c)`. Each mixture takes a prefix of this same source-specific ordering with the count in Table 6. Thus different workloads may share source episodes. Each target uses a fixed LSS assembled from its sources' selection-order prefixes, averaging relevance across all selected support episodes. Source membership identifies an available source anchor during preparation; per-request source identity is not supplied at test time.

Base and LoraRetriever can reuse predictions across targets when model, adapter pool, episode input, history, and decoding configuration are identical. LoGo mixture results reuse the recorded single-client predictions and therefore retain the routing context of those source runs; they do not replay the selected episodes as a new stateful sequence. GUIHub and LoRAHub instead evaluate each mixture using its own fixed composition. The Sensitivity group tests these fixed proportions rather than all possible mixture weights.

B.2 Supplementary cross-client results

Tables 7–8 report Type, Ground, and Step SR for Primary, Extension, Sensitivity, and All on both Qwen backbones. Semantic Top-1 is an additional Qwen3.5-4B retrieval diagnostic, with IID/OOD Step SR of 46.70%/49.36%; it changes both selection evidence and single-adapter execution, so it is separate from the matched semantic-routing ablation.

B.3 Hub expansion and initially covered clients

For each backbone, the full candidate collection contains 32 independently trained adapters. Each stage uses a prefix of the arrival order, sorted from lower to higher mean GSS response, without using test outcomes. The shared capability basis is fitted to the 24 IID reference adapters before

Table 7: Qwen3.5-4B: supplementary mixed-client accuracy (%). All metrics use pooled correct/total counts; the All row combines all 16 mixtures.

Method	Split	Type	Ground	Step SR
Base	Primary	40.79	9.78	22.88
Base	Extension	43.68	10.97	24.18
Base	Sensitivity	42.18	8.96	24.24
Base	All	41.84	9.95	23.50
LoRAHub	Primary	79.99	31.64	44.83
LoRAHub	Extension	82.54	37.49	49.12
LoRAHub	Sensitivity	80.25	29.06	44.75
LoRAHub	All	80.73	32.76	45.98
LoGo	Primary	79.05	58.58	61.18
LoGo	Extension	82.34	64.22	64.74
LoGo	Sensitivity	77.10	55.49	59.16
LoGo	All	79.57	59.54	61.76
LoraRetriever	Primary	79.43	59.62	62.15
LoraRetriever	Extension	82.54	64.43	65.56
LoraRetriever	Sensitivity	77.77	56.57	60.21
LoraRetriever	All	79.96	60.36	62.70
GUIHub	Primary	81.44	62.35	64.71
GUIHub	Extension	84.85	67.41	69.29
GUIHub	Sensitivity	81.11	58.27	62.98
GUIHub	All	82.30	62.97	65.62
Semantic Top-1	Primary	80.68	30.98	44.31
Semantic Top-1	Extension	80.77	36.32	46.26
Semantic Top-1	Sensitivity	80.73	27.67	44.18
Semantic Top-1	All	80.71	31.82	44.82

Table 8: Qwen3.5-9B: supplementary mixed-client accuracy (%). All metrics use pooled correct/total counts; the All row combines all 16 mixtures.

Method	Split	Type	Ground	Step SR
Base	Primary	51.05	14.70	25.92
Base	Extension	52.99	12.89	26.15
Base	Sensitivity	53.63	15.92	27.58
Base	All	52.08	14.43	26.31
LoRAHub	Primary	78.50	53.17	57.03
LoRAHub	Extension	82.54	60.70	64.61
LoRAHub	Sensitivity	78.63	54.71	59.16
LoRAHub	All	79.62	55.53	59.50
LoGo	Primary	79.57	64.70	63.98
LoGo	Extension	82.54	68.69	67.26
LoGo	Sensitivity	78.82	64.14	63.65
LoGo	All	80.23	65.69	64.81
LoraRetriever	Primary	80.68	65.03	64.85
LoraRetriever	Extension	82.47	68.37	67.53
LoraRetriever	Sensitivity	80.15	63.99	64.41
LoraRetriever	All	81.06	65.75	65.49
GUIHub	Primary	80.50	69.29	68.58
GUIHub	Extension	84.65	73.27	72.76
GUIHub	Sensitivity	79.58	73.72	70.42
GUIHub	All	81.45	71.22	70.07

Table 9: Initially covered clients: each backbone fixes the eight targets whose adapters are present in the initial eight-adapter Hub and evaluates the same 400 episodes at every stage. Step SR (%) pools action counts. The last column counts clients improved/unchanged/decreased as the Hub expands from 8 to 32 adapters, using unrounded counts.

Backbone	Method	8 adapters	32 adapters	Δ (pp)	$+/=/-$
Qwen3.5-4B	GUIHub	60.47	67.34	+6.87	8/0/0
	LoraRetriever	60.83	62.89	+2.07	6/0/2
	LoGo	60.47	60.00	-0.47	3/2/3
	LoRAHub	41.40	45.89	+4.50	6/0/2
Qwen3.5-9B	GUIHub	68.88	71.54	+2.66	5/1/2
	LoraRetriever	65.10	65.97	+0.87	6/1/1
	LoGo	62.75	63.46	+0.72	4/1/3
	LoRAHub	54.01	58.41	+4.39	4/0/4
Gemma-4-E4B-it	GUIHub	48.02	48.57	+0.55	4/0/4
	LoraRetriever	43.97	46.87	+2.90	6/0/2
	LoGo	38.51	41.11	+2.60	5/1/2
	LoRAHub	40.56	41.01	+0.45	4/1/3

the evaluated adapter pool begins to expand. Consequently, expansion measures reuse under a fixed representation, not construction of a capability space from only the initially admitted eight adapters.

The orders are as follows (client IDs):

- Qwen3.5-4B: 27, 15, 4, 9, 10, 13, 12, 16, 7, 26, 24, 17, 31, 2, 18, 22, 30, 20, 11, 19, 3, 21, 23, 6, 29, 0, 5, 14, 25, 1, 8, 28.
- Gemma-4-E4B-it: 11, 9, 21, 13, 27, 3, 14, 24, 26, 17, 31, 16, 6, 25, 4, 20, 1, 28, 2, 18, 19, 8, 7, 10, 22, 12, 23, 15, 30, 29, 0, 5.
- Qwen3.5-9B: 9, 11, 16, 29, 17, 18, 14, 24, 25, 13, 12, 27, 6, 4, 31, 21, 2, 22, 5, 7, 1, 15, 30, 26, 20, 28, 19, 3, 0, 10, 8, 23.

The first eight entries identify initially covered target clients. They include seven IID and one OOD client for each Qwen backbone, and six IID and two OOD clients for Gemma. When a target's own adapter becomes available, it can serve as the anchor; otherwise selection remains anchor-free. IID/OOD labels remain the original evaluation groups and do not imply that their adapters stay absent at every expansion stage. Table 9 reports results for this fixed initial cohort, evaluated using 400 test episodes per backbone at every stage. As the Hub expands from 8 to 32 adapters, GUIHub's pooled Step SR increases by 6.87, 2.66, and 0.55 pp on Qwen3.5-4B, Qwen3.5-9B, and Gemma, respectively. The corresponding numbers of improved/unchanged/decreased clients are 8/0/0, 5/1/2, and 4/0/4. Aggregate improvement therefore coexists with individual regressions, and intermediate stages need not improve monotonically.

C Efficiency Breakdown

Workload and execution. The measured workload contains 32 targets, with five episodes at zero-based positions 0, 12, 25, 37, and 49 of each target's fixed test ordering. The resulting 160 episodes contain 998 measured action executions. Each method runs three repeats, retaining all requests in the selected episodes. The 24-adapter pool is the first 24 entries of the Qwen3.5-4B expansion order, which differs from the 24-adapter IID pool used in the main accuracy experiments. The run uses A100-SXM4-80GB GPUs, driver 570.133.20, PyTorch 2.9.1, Transformers 5.12.1, ms-swift 4.5.2, and PEFT 0.19.1, with two CPU threads per inference process. Generation uses BF16, batch size one, greedy decoding with reasoning output disabled, seed 42, and a 128-token budget. LoRAHub's objective is evaluated in microbatches of two examples.

Recorded time. Each operation is timed using a monotonic wall clock, with GPU synchronization immediately before and after it, so its duration includes CPU computation and time spent waiting for the GPU. For repeat r, let P_r, R_r, and E_r be preparation, target setup, and model initialization, and

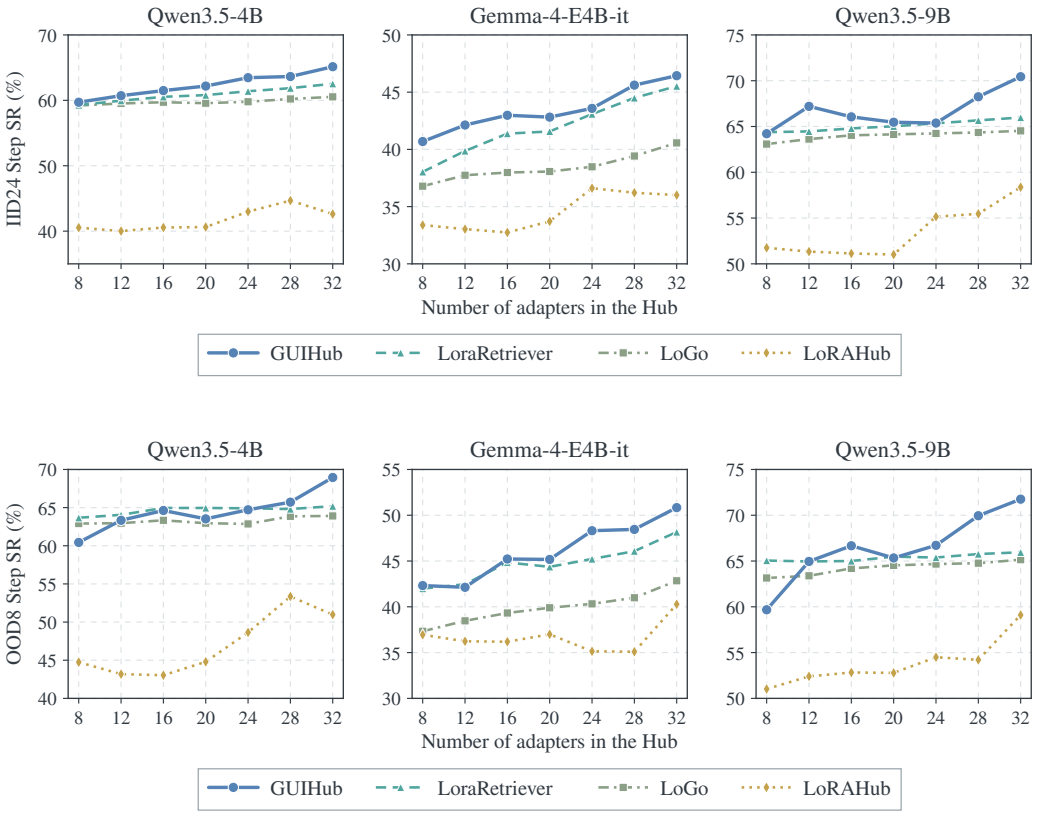


Figure 8: IID and OOD Step SR (%) during Hub expansion. Each curve pools correct/total counts over a fixed test group at each stage: 1,200 IID episodes and 400 OOD episodes. Top: IID; bottom: OOD. These labels identify fixed target groups. Each panel uses a separate vertical range. No error bars are estimated.

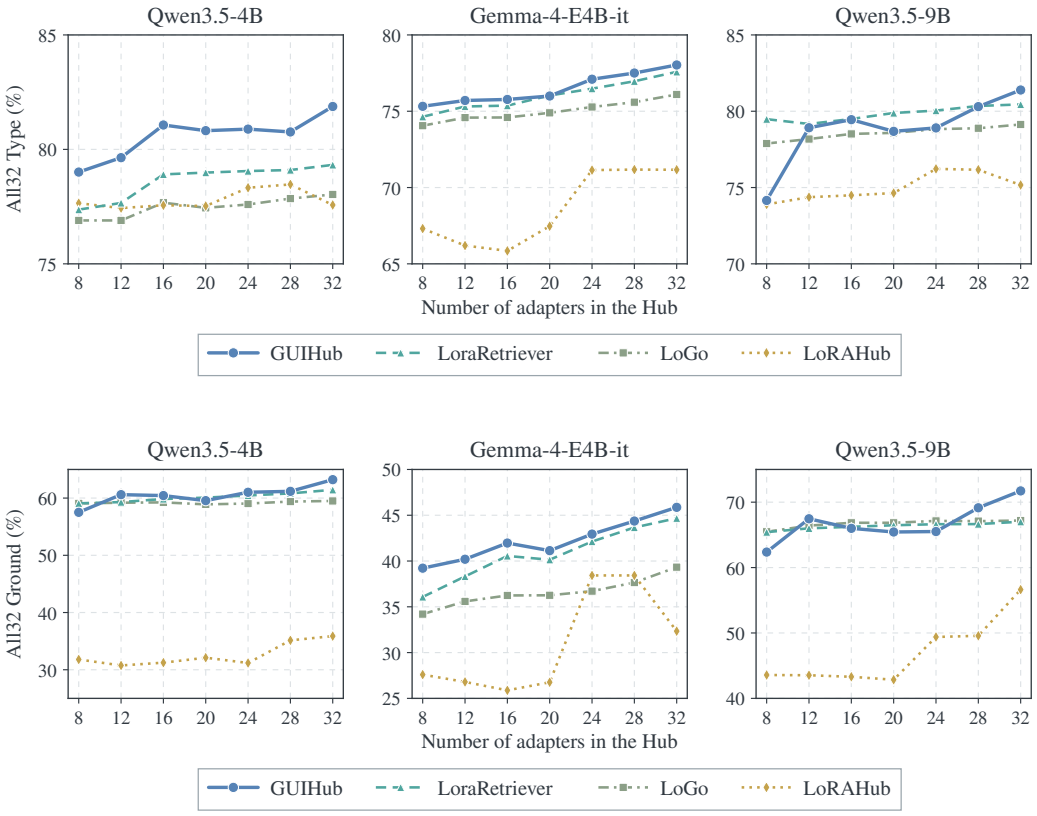


Figure 9: Type and Ground Accuracy across all 32 clients (%) during expansion. Type pools over 8,899 valid non-terminal steps; Ground uses only grounding-applicable steps. All stages use the same 1,600 test episodes. Top: Type; bottom: Ground. Each panel uses a separate vertical range. No error bars are estimated.

Table 10: Recorded timing means ± sample standard deviations across three repeats of 160 episodes/998 actions. Total is setup plus online execution. Online latency and throughput include LoraRetriever retrieval/fusion and LoGo routing; throughput is actions/s. Statistics are computed from repeat-level values. Timing boundaries are method-specific as described in this appendix.

Method	Setup (s)	Online (s/action)	Total (h)	Throughput
LoRAHub	5058.732 ± 6.186	4.102 ± 0.002	2.542 ± 0.002	0.244 ± 0.000
LoraRetriever	151.613 ± 0.001	12.790 ± 0.868	3.588 ± 0.241	0.078 ± 0.005
LoGo	27.189 ± 3.899	24.726 ± 1.829	6.862 ± 0.508	0.041 ± 0.003
GUIHub	54.903 ± 3.406	4.354 ± 0.292	1.222 ± 0.082	0.230 ± 0.015

Table 11: Mean cumulative timing components per repeat (seconds). The routing/fusion column reports a component of online execution for LoraRetriever and LoGo, already included in their totals. LoRAHub model loading is included in preparation and target setup.

Method	Preparation (s)	Target setup (s)	Model initialization (s)	Routing/fusion (s)
LoRAHub	4902.348	156.384	0.000	0.000
LoraRetriever	0.074	0.000	151.539	40.317
LoGo	0.000	0.000	27.189	11428.973
GUIHub	0.089	9.566	45.248	0.000

let O_r be online execution. For LoraRetriever, $O_r = I_r + F_r$, where I_r is the recorded inference time and F_r is the separately recorded retrieval/fusion time; these intervals do not overlap. Figure 4 uses

$$S_r = P_r + R_r + E_r, \qquad T_r = S_r + O_r. \tag{10}$$

The bars show $\overline{O}/3600$ and $\overline{S}/3600$ hours. Error bars are the sample standard deviation of $T_r/3600$ across three repeats, not the sum of component deviations or a confidence interval. Online latency and throughput are computed as $O_r/998$ and $998/O_r$ for each repeat, then summarized by their means and sample standard deviations. Speed ratios divide mean recorded totals before rounding.

Method-specific boundaries. LoRAHub preparation includes local objective search and the model loading needed for that search; target setup includes constructing the composed adapter and loading the model for each target. Its separately reported model-initialization time is zero because loading is already included in these two measurements. GUIHub preparation loads precomputed routing information, model initialization loads the backbone and adapters, and target setup constructs and applies one composition for each of the 32 targets. The saved measurements do not separate adapter selection from applying the resulting composition.

LoraRetriever's separately recorded query retrieval and factor fusion average 40.317 seconds per repeat and are included in online execution and total time. This measurement retains repeated retrieval and composition activation for the first episode of targets 1–31; individual operation durations were not saved, so no estimated correction is subtracted. LoGo's online time already includes routing and generation, so its separately reported routing time is a breakdown component and is not added again. LoRAHub and GUIHub include target-level adapter selection in setup time.

Warmup, hardware, and exclusions. LoraRetriever, LoGo, and GUIHub keep one model loaded throughout each repeat and warm up on the first three requests. LoRAHub loads a model for each target and warms up on up to three requests of that target's first episode (95 warmup calls per repeat for the selected episodes). Warmup is excluded from recorded online time; LoGo retains its selected adapters and weights after warmup and between episodes. LoRAHub was measured earlier on different physical GPUs of the same model. The four methods were therefore not measured in paired runs on the same devices. GPU clock speeds and operating-system and filesystem caches were not controlled. Recorded totals exclude work outside timed intervals, cleanup, output writing, scheduling, and the offline computation described below; total elapsed time for the complete evaluation process and generated-token counts were not recorded.

Offline preparation. Timing starts from trained adapters and precomputed assets: GSS behavioral profiles, capability vectors and their projection basis, and semantic embeddings of support episodes

and test queries. Their construction costs are excluded and are not estimated here. LoRAHub's target-specific objective search is included in preparation.

D Additional Ablations and Diagnostics

D.1 Benefit of adapter collaboration

We compare GUIHub with Own-LoRA, which directly loads each target client's corresponding adapter, on Qwen3.5-4B using the same 50 test episodes per client, decoder, teacher-forced history, and evaluator.

Table 12: Benefit of adapter collaboration on Qwen3.5-4B. Step SR (%, ↑) uses pooled counts. Own-LoRA directly loads the target's adapter.

Method	All	IID	OOD
GUIHub	65.46	64.98	67.00
Own-LoRA	47.36	46.90	48.88

GUIHub outperforms directly loading the corresponding LoRA, improving Step SR by 18.09, 18.08, and 18.12 pp on all 32 clients, the 24 IID clients, and the 8 OOD clients, respectively (Table 12).

D.2 Component ablations

We evaluate three component ablations on Qwen3.5-4B, each changing only the component specified below while retaining all other settings of the main GUIHub configuration. Semantic routing and Direct cover the same 1,200 IID episodes, 400 OOD episodes, and 960 episode records across 16 mixed targets; the no-anchor variant is independently evaluated on IID and mixed targets, with its OOD results reused from the main GUIHub experiment. Table 13 reports accuracy from pooled correct and eligible step counts within each setting.

Semantic routing. We replace behavioral compatibility with semantic relevance. Let $\mathcal{G}_i$ contain the 32 frozen GSS episodes associated with adapter i's source client, and let $\widehat{e}_{u,\ell}$ and $\widehat{e}^{\mathrm{G}}_j$ denote unit-normalized LSS and GSS embeddings. The score is

$$s^{\mathrm{sem}}_{u,i} = \frac{1}{|\mathcal{L}_u|\,|\mathcal{G}_i|} \sum_{\ell \in \mathcal{L}_u} \sum_{j \in \mathcal{G}_i} \widehat{e}^{\top}_{u,\ell} \widehat{e}^{\mathrm{G}}_j. \tag{11}$$

This score replaces $s_{u,i}$ in anchor/helper selection and the selected-set softmax, without additional normalization. Routing uses neither behavioral outcomes nor test queries.

Without anchor. We remove the own/source anchor constraint and select the two highest-scoring adapters, retaining the behavioral scores and composition rule. GUIHub already uses anchor-free selection for OOD targets in RQ2. We therefore reuse its Qwen3.5-4B results under the same configuration (Table 1); the OOD row is not an additional independent measurement. The IID and mixed-client results here and the corresponding anchor-free sensitivity reference in Appendix D.3 come from two independent inference runs under the same configuration; each analysis retains its own reference results.

Direct matching. We replace the projected compatibility score with $s^{\mathrm{Direct}}_{u,i} = \sum_j \rho_u(j) R_{ij}$. Only the routing projection is removed; the frozen GSS construction, including its use of SVD, is retained. The full evaluation includes 640 reused predictions from the exploratory pilot.

The projected score can be interpreted as a relevance-weighted reconstructed response up to a target-specific offset. Define

$$\mathbf{r}_i = R^{\top}_{i,:}, \qquad \hat{\mathbf{r}}_i = \mu + BB^{\top}(\mathbf{r}_i - \mu),$$

where $\hat{\mathbf{r}}_i$ is the reconstruction in the shared SVD subspace. Using $c_i = B^{\top}(\mathbf{r}_i - \mu)$ and $q_u = B^{\top}\rho_u$, we obtain

$$\begin{aligned} s_{u,i} &= c_i^{\top} q_u = \rho_u^{\top} BB^{\top}(\mathbf{r}_i - \mu) \\ &= \rho_u^{\top} \hat{\mathbf{r}}_i - \rho_u^{\top} \mu. \end{aligned}$$

Table 13: Component ablations on Qwen3.5-4B. Accuracy (%) is computed from pooled counts within each setting. Mixed (All) includes all 16 mixed targets. The no-anchor OOD row (†) reuses the main GUIHub results under the same configuration, since RQ2 already uses anchor-free selection; it is not an additional independent measurement.

Variant	Split	Type	Ground	Step SR
GUIHub	IID	81.71	63.08	64.98
GUIHub	OOD	81.84	61.31	67.00
GUIHub	Mixed (All)	82.30	62.97	65.62
Semantic routing	IID	81.72	61.60	63.96
Semantic routing	OOD	80.98	54.06	61.82
Semantic routing	Mixed (All)	81.73	58.20	61.76
Without anchor	IID	81.53	61.71	63.99
Without anchor	OOD†	81.84	61.31	67.00
Without anchor	Mixed (All)	82.73	61.56	65.16
Direct	IID	81.72	63.53	65.19
Direct	OOD	82.88	61.57	67.76
Direct	Mixed (All)	82.73	62.94	65.88

Here ρ_u assigns task-relevance weights to GSS probes; it is not another adapter response, so the response mean μ is not subtracted from it. For a fixed target and shared μ and B, the offset $\rho_u^\top \mu$ is common to all candidates. Consequently, $s_{u,i}$ and $\rho_u^\top \hat{\mathbf{r}}_i$ induce the same adapter ordering and the same softmax weights over a fixed selected set at the same routing temperature.

Define the residual $\epsilon_i = (I - BB^\top)(\mathbf{r}_i - \mu)$, so that $\mathbf{r}_i = \hat{\mathbf{r}}_i + \epsilon_i$. Direct matching then satisfies

$$s_{u,i}^{\text{Direct}} = \rho_u^\top \mathbf{r}_i = s_{u,i} + \rho_u^\top \mu + \rho_u^\top \epsilon_i.$$

Any new adapter profiled on the same GSS can be mapped using the frozen μ and B. The term $\rho_u^\top \epsilon_i$ captures the contribution of omitted response directions to its Direct score and can vary across adapters; mapping into the fixed space does not guarantee lossless representation of these directions.

The SVD-based projection provides compact capability descriptors while retaining Step SR close to Direct matching in the evaluated settings. It maps each adapter's P-dimensional behavioral response to K dimensions, with the projection matrix B and mean μ shared across adapters within each backbone. With $P = 768$ and the default $K = 8$, each capability descriptor stores $1/96$ as many scalar values (96:1 compression); LoRA weights are not compressed. For N adapters, descriptor storage is $O(NK)$ plus $O(PK + P)$ shared storage, compared with $O(NP)$ for dense response descriptors. Given the target relevance vector ρ_u, computing $q_u = B^\top \rho_u$ costs $O(PK)$ once per target, followed by $O(NK)$ candidate scoring, versus $O(NP)$ for dense Direct matching. These are theoretical storage and arithmetic properties, not measured system speedups. Candidate scoring requires only the capability vectors and target demand, without reading the full behavioral response vectors or candidate LoRA weights. A lightweight routing service could therefore maintain a compact index separately from the weight repository and load only selected adapters for execution; this is a deployment possibility, not a system benefit demonstrated at scale in this work. Empirically, projected matching achieves Step SR values only 0.21, 0.76, and 0.26 pp below Direct on IID, OOD, and Mixed (All), respectively (Table 13), supporting a compact representation with a small observed accuracy trade-off.

D.3 Hyperparameter sensitivity

We evaluate four hyperparameters on Qwen3.5-4B, changing one at a time from the reference values $k = 2$, the standard LSS budget, $K = 8$, and $M = 64$. All conditions use the same 768-episode GSS, routing temperature 0.1, relevance temperature 0.07, composition scale 1.5, checkpoints, decoder, and pooled evaluator, with 1,200 IID episodes, 400 OOD episodes, and 960 mixed-client episode records. Adapter-count, support-budget, and capability-dimension comparisons use anchor-free selection; relevance-budget comparisons retain the main GUIHub anchor/helper policy. Table 14 provides a separate reference for each selection policy. For IID and mixed clients, the anchor-free reference here and the no-anchor component ablation in Appendix D.2 come from two independent inference runs under the same configuration; each analysis retains its own reference results.

Table 14: Hyperparameter sensitivity on Qwen3.5-4B. Accuracies are percentages (↑) from pooled counts. Each variant changes one parameter relative to the reference within its selection-policy group. Standard LSS uses 10 episodes for single-client targets and Primary/Sensitivity mixtures, and 9 for Extension; the reduced budget uses 5 for every target. LoRA training rank stays 16.

Setting	k	LSS budget	K	M	Split	Type	Ground	Step SR
Anchor-free selection								
Reference	2	Standard	8	64	IID	81.50	61.85	64.05
					OOD	81.84	61.31	67.00
					Mixed (All)	82.73	61.59	65.18
Adapter count	1	Standard	8	64	IID	80.49	61.44	63.43
					OOD	81.17	61.66	67.24
					Mixed (All)	81.71	61.48	64.83
Adapter count	3	Standard	8	64	IID	81.64	62.17	64.20
					OOD	82.36	61.66	67.48
					Mixed (All)	82.49	60.95	64.38
Support budget	2	5	8	64	IID	81.39	62.21	64.23
					OOD	82.31	61.40	67.33
					Mixed (All)	82.75	61.77	64.92
Capability dimension	2	Standard	4	64	IID	81.37	60.76	63.38
					OOD	81.17	60.70	66.29
					Mixed (All)	81.90	60.77	64.23
Capability dimension	2	Standard	16	64	IID	81.52	62.83	64.52
					OOD	82.64	61.22	67.33
					Mixed (All)	83.08	62.76	65.82
Main GUIHub anchor/helper policy								
Reference	2	Standard	8	64	IID	81.71	63.08	64.98
					OOD	81.84	61.31	67.00
					Mixed (All)	82.30	62.97	65.62
Relevance budget	2	Standard	8	768	IID	81.86	62.90	64.96
					OOD	81.98	61.31	67.05
					Mixed (All)	82.43	62.70	65.53

Adapter count. We vary the number of composed adapters over $k \in \{1, 2, 3\}$, selecting the highest-scoring candidates under the anchor-free rule.

Support budget. We compare the standard LSS budget with five episodes per target. The standard budget contains 10 episodes for each single-client target and for Primary/Sensitivity mixtures, and 9 for Extension mixtures.

Capability dimension. We vary the centered behavioral capability dimension over $K \in \{4, 8, 16\}$. LoRA training rank remains 16 in every condition.

Relevance budget. We compare $M = 64$ with $M = 768$ relevant GSS probes per LSS episode. At $M = 768$, softmax covers the entire GSS; the per-LSS relevance distributions are averaged as in the reference, retaining the centered $K = 8$ representation.

Across the tested settings and splits in Table 14, the absolute Step SR change is less than 1 pp relative to the corresponding reference. Reducing the LSS budget to five episodes changes Step SR by at most 0.33 pp. Removing relevance truncation changes it by -0.01, $+0.05$, and -0.09 pp on IID, OOD, and Mixed (All), respectively, all below 0.10 pp in magnitude. These results indicate limited sensitivity within the tested settings, although effects differ across target distributions.

D.4 Routing scores and selected-set capability

We evaluate 768 target–adapter combinations on Qwen3.5-4B: 32 targets, 24 adapters, and 50 test episodes per combination. Each singleton has weight one and composition scale $\gamma = 1.5$ in addition

Table 15: Mean per-target score correlations and selected-set regret. Brackets are 95% target-bootstrap intervals (10,000 resamples) for regret; correlation and regret are descriptive diagnostics. All, IID, and OOD contain 32, 24, and 8 targets, respectively.

Split	Signal	Spearman	Pearson	Regret (pp)
All	Behavior	0.513	0.520	1.09 [0.60, 1.64]
All	Semantic	0.103	0.147	3.07 [1.93, 4.33]
IID	Behavior	0.498	0.503	0.98 [0.44, 1.60]
IID	Semantic	0.145	0.214	1.95 [1.03, 3.02]
OOD	Behavior	0.557	0.572	1.43 [0.46, 2.63]
OOD	Semantic	-0.024	-0.053	6.45 [3.76, 9.31]

Table 16: Paired whole-episode bootstrap for Step SR differences (pp). Clients are fixed within each test group; 50 complete episodes are resampled per client, with paired draws across compared conditions. Intervals are pointwise 95% percentile intervals from 10,000 replicates (seed 42).

Comparison / backbone	IID Δ	95% CI	OOD Δ	95% CI
GUIHub − LoraRetriever				
Qwen3.5-4B	+2.74	[+1.55, +3.94]	+2.57	[+0.38, +4.73]
Qwen3.5-9B	+4.21	[+3.04, +5.35]	+5.61	[+3.55, +7.66]
Gemma-4-E4B-it	+1.75	[+0.78, +2.73]	+0.90	[-0.87, +2.72]
Behavioral − Semantic (4B)	+1.02	[+0.14, +1.87]	+5.18	[+3.50, +6.84]

to its trained LoRA scaling α/r. The frozen selected sets are those of behavioral and matched semantic routing. Regret compares the best singleton among a selected set with the best singleton in the entire pool; it neither uses composition weights nor evaluates an alternative pair by combining its adapters.

Correlations are computed separately within each target over 24 candidates, then averaged. Target bootstrap intervals use 10,000 resamples and are descriptive, especially for the eight OOD targets. The behavioral Spearman mean across all 32 clients is 0.513 with 95% interval $[0.426, 0.591]$; its Pearson mean is 0.520 with interval $[0.435, 0.599]$. Behavioral selected-set regret is 1.09 pp with interval $[0.60, 1.64]$, versus 3.07 pp with interval $[1.93, 4.33]$ for semantic routing. The selected set contains the pool-best singleton for 56.25% of targets under behavioral routing and 34.38% under semantic routing. Mean overlap with the two best singletons is 0.844 and 0.563 adapters, respectively. These statistics diagnose retrieval quality, not composition optimality.

Figure 5 visualizes score alignment and paired selected-set regret in the main text.

D.5 Paired-episode bootstrap analysis

IID/OOD comparisons (RQ1–RQ2). For the IID/OOD results, we estimate test-sampling uncertainty for GUIHub versus LoraRetriever on all three backbones and for behavioral versus strictly matched semantic routing on Qwen3.5-4B. We fix the set of 24 IID or 8 OOD clients and resample 50 complete test episodes with replacement within each client. All compared methods share the sampled episode IDs. For each replicate, we sum action-level correct/total counts and compute the difference in pooled Step SR; individual steps are not resampled independently and episode success rates are not averaged.

We use 10,000 replicates with seed 42 and report the 2.5th/97.5th percentile endpoints with linear interpolation. Within each split, clients and episode IDs are sorted, the random seed is reset, and all methods/backbones share the draws. The IID/OOD evaluation sets contain 1,200/400 episodes and 6,796/2,103 valid steps, respectively. Gemma uses the main-table scale of 1.0; the semantic comparison uses only the matched routing-score variant.

Table 16 shows positive GUIHub–LoraRetriever intervals on IID for all three backbones and on OOD for both Qwen backbones. Gemma's OOD interval includes zero. The behavioral–semantic intervals

Table 17: Paired-episode bootstrap for Step SR differences across all 32 clients during Hub expansion (pp). Intervals are pointwise 95% percentile intervals from 10,000 replicates (seed 42).

	GUIHub 32 vs. 8 adapters		32 adapters GUIHub − LoraRetriever	
Backbone	Δ	95% CI	Δ	95% CI
Qwen3.5-4B	+6.16	[+5.24, +7.09]	+2.88	[+1.86, +3.87]
Qwen3.5-9B	+7.61	[+6.75, +8.46]	+4.78	[+3.73, +5.81]
Gemma-4-E4B-it	+6.41	[+5.44, +7.39]	+1.33	[+0.48, +2.16]

exclude zero on both splits; the larger OOD point estimate does not by itself establish an IID/OOD effect difference.

Hub expansion (RQ4). We apply the same paired-episode bootstrap to two Step SR differences across all 32 clients: GUIHub with 32 versus 8 adapters, and GUIHub minus LoraRetriever with a 32-adapter Hub (Figure 3). The 32 clients are fixed, with 50 episodes per client (1,600 episodes and 8,899 valid steps). All compared conditions share the sampled episode IDs.

Table 17 reports positive pointwise 95% percentile intervals for all six Hub-expansion endpoint differences. These intervals describe test-sampling uncertainty conditional on fixed trained models, selected configurations, clients, and each backbone's recorded arrival order.