CLASP: Retrofitting Cost–Latency SLO Compliance onto Any LLM Router

Hak Hyun Kim [1] **Benjamin Huh** [1] **Jimin Moon** [2]
Chia-Wei Lee [1] **Jason Peng** [1] **Wesley J. Marrero** [1] **Soroush Vosoughi** [1]

Abstract

LLM routers are typically trained to maximize answer quality, ignoring the *hard* per-query cost and latency Service Level Objectives (SLOs) that production imposes on both resources at once; these targets also shift, and the deployed cost/latency distribution can drift from any offline measurement, so a router fixed at training time goes stale. We introduce **CLASP**, a *post-hoc* controller that retrofits cost–latency SLO compliance onto *any* router emitting per-action quality scores, without retraining it: it masks chronically infeasible actions, subtracts its own query-adaptive cost/latency penalties, and binds to a new SLO by recomputing just a feasibility mask and two penalty weights. Across five mechanism-diverse published routers, CLASP cuts cost and latency violations (as high as 51%) to the 5% target at the moderate and relaxed SLOs and sharply reduces them at the tightest, while keeping quality within 1.4 pp for strong bases and, for weak ones, raising it as the feasibility filter discards their costly-and-weak picks. And when the deployed distribution drifts (a load spike that slows, or a price change that raises the cost of, the router's preferred actions), CLASP's online closed-loop adapts from live outcomes and recovers compliance (violations fall from 38–54% to 2%) where a static controller stays broken. A controlled sweep further exposes a *compliance–quality frontier*: reaching compliance costs essentially no quality at loose SLOs but must be paid for in quality at the tightest, which CLASP makes explicit and navigable.

[1]Dartmouth College, Hanover, NH, USA [2]Columbia University, New York, NY, USA. Correspondence to: Soroush Vosoughi <Soroush.Vosoughi@dartmouth.edu>.

Accepted at ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning. Copyright 2026 by the author(s).

1. Introduction

A large and growing family of LLM routers learns, for each query, which action to take: a single model (Ong et al., 2024; Ding et al., 2024), a latent-ability profile (Song et al., 2025; Zhuang et al., 2025), or a multi-agent topology (Yue et al., 2025; Zhang et al., 2025), so as to maximize answer quality, sometimes traded off against a *soft* cost term. Two facts about real deployments sit uneasily with this design. First, serving systems impose *hard* Service Level Objectives (SLOs) on *both* cost and latency: a fraction of requests above a per-query budget or deadline is a contract breach, not a tunable penalty. Second, these targets *shift*: latency may tighten from 90 s to 60 s during peak hours and relax off-peak (Chen et al., 2023), so a router fixed at training time cannot track them, and retraining a fresh policy for every regime is impractical.

Recent SLO-aware routers do enforce constraints, but they bake the constraint machinery *into* a bespoke trained policy: PROTEUS conditions a Lagrangian-RL policy on a runtime accuracy target (Bhatti et al., 2026), and bandit routers condition a single policy on a preference/SLO vector learned during training (Wang et al., 2025). Such designs are powerful but monolithic: the constraint logic is inseparable from one particular router and one constraint formulation, and cannot be reused to make the many existing quality-only routers SLO-compliant.

We take the opposite, modular stance. We introduce **CLASP** (Cost–Latency-Aware SLO-Preserving control), a *post-hoc* controller that retrofits cost–latency SLO compliance onto *any* base router that emits a per-action quality score, *without retraining the base*. For a query, CLASP (i) masks actions whose training-set violation rate exceeds a feasibility threshold; (ii) predicts the query's per-action cost and latency with its own lightweight models (trained offline from the same cached outcomes, independent of the base) and converts them into penalties against the active SLO; and (iii) drives the realized violation rate to a target ε by online dual ascent on two scalars. Because the SLO thresholds enter only at this control layer, adapting to a new SLO re-runs the controller on a small calibration set; and when the *deployed* cost/latency distribution drifts away from the offline

cache, CLASP updates its per-action estimates online from observed outcomes so it tracks the drift, all with no base retraining and no access to the base's parameters.

We evaluate CLASP by retrofitting it onto five *mechanism-diverse* published routers (latent-factor IRT (Song et al., 2025), matrix factorization (Zhuang et al., 2025), dual contrastive embedding (Chen et al., 2024), similarity-weighted kNN (Ong et al., 2024), and a VAE topology controller (Yue et al., 2025)), reimplemented on a shared 33-action multi-agent topology space with cached cost, latency, and quality. Each base, run alone, violates the SLOs (by up to 51%, though two of the five are already near compliance under loose SLOs); with CLASP attached and no base retraining, violations fall to the 5% target at the moderate and relaxed SLOs and are sharply reduced at the tightest, with the feasibility filter raising quality for weak bases (by discarding their costly-and-weak picks) and quality changed by at most 1.4 pp for strong ones. Probing further, we find a *compliance–quality frontier*: at loose and moderate SLOs full compliance costs essentially no quality, but at the tightest SLOs the feasible high-quality actions genuinely violate, so compliance must be bought with quality, which CLASP exposes through a single feasibility knob. Finally, when we inject a deployment drift on the router's preferred actions, unseen by the offline cache, in latency (a load spike) and in cost (a price increase), a static, offline-only controller stays in violation (38–54%) while CLASP's online closed loop recovers compliance (2%): the online machinery earns its keep precisely when deployment departs from the offline data.

Our contributions:

1. **CLASP, a post-hoc cost–latency SLO controller.** A base-agnostic module (feasibility filter, query-adaptive cost/latency predictors, and an online violation-rate closed loop) that brings *any* per-action-scoring router into joint cost and latency SLO compliance at the moderate and relaxed regimes (sharply cutting violations at the tightest), and transfers across regimes with no base retraining. Unlike constraint-in-training routers (Bhatti et al., 2026; Wang et al., 2025), the constraint logic is modular and reusable.
2. **Base-agnostic across diverse mechanisms.** We retrofit CLASP, unchanged, onto reimplementations of five published routers spanning five distinct routing mechanisms (re-expressed on a shared action space and supervised by the same F1 cache); all become compliant at the moderate and relaxed SLOs, with violations sharply reduced at the tightest, the feasibility filter raising quality for weak bases and quality within 1.4 pp for strong ones. We do not claim universality beyond these five, but CLASP attaches to any router by construction.
3. **Online adaptation from offline data to deployment drift.** When the deployed cost or latency distribution shifts away from the offline cache, CLASP's online closed loop (refreshing its per-action estimates as its penalty weights ascend) recovers compliance under both latency and cost drift across all five bases (final windowed violation 0–8%, within ε except two residual endpoints at 6–8%), where a static, offline-only controller stays broken: the offline-to-online setting that the method targets. Under static conditions the offline filter alone suffices; under drift the full loop is what recovers, and both updates are typically needed on the hardest-hit bases.

2. Related Work

Multi-agent topology routing. Recent multi-agent systems select per-query *collaboration topologies*: MasRouter learns a cascaded controller over collaboration modes and roles (Yue et al., 2025), MaAS samples query-conditioned architectures from an agentic supernet (Zhang et al., 2025), G-Designer designs communication graphs with VGAEs (Zhang et al., 2024), and DAAO orchestrates difficulty-aware workflows (Su et al., 2025). These optimize quality and token efficiency as soft objectives and do not enforce hard cost/latency SLO feasibility or adapt across shifting regimes. Our action space (33 topology$\times$tier configurations) follows this line, and MasRouter is one of our base routers.

Quality-oriented LLM routing. Single-model routers select among LLMs to balance quality and cost: RouteLLM learns strong/weak routers from preference data via matrix factorization, similarity-weighted ranking, or classifiers (Ong et al., 2024); HybridLLM trains a difficulty classifier (Ding et al., 2024); and FrugalGPT cascades from cheaper to more expensive models (Chen et al., 2023). A second line models the query–LLM interaction with latent factors: IRT-Router fits multidimensional Item Response Theory to predict per-LLM accuracy and routes by a performance-minus-cost score (Song et al., 2025); EmbedLLM learns compact model embeddings via matrix factorization for correctness prediction and routing (Zhuang et al., 2025); and RouterDC trains a query encoder and per-LLM embeddings with a dual contrastive objective, routing by embedding similarity (Chen et al., 2024). All of these optimize quality (optionally against a soft cost term) and *none* enforces a hard latency or violation-rate SLO; standardized benchmarks likewise score routers on quality and cost alone (Hu et al., 2024). CLASP treats such routers as interchangeable base routers and supplies the missing constraint layer.

Constraint-aware routing. Closest to our goal are routers that handle explicit constraints, but they embed the constraint into training. PROTEUS trains a Lagrangian-RL policy conditioned on a runtime accuracy target so one model serves an accuracy spectrum (Bhatti et al., 2026), BaRP conditions a contextual-bandit policy on a preference vector for test-time trade-offs (Wang et al., 2025), and OmniRouter solves for budget/performance-controllable trade-offs (Mei et al., 2025). These yield a single constrained policy but are not modular: the controller cannot be detached and reused on a different base, they target a single budget rather than simultaneous cost *and* latency violation-rate SLOs, and PROTEUS fixes its dual variable after training rather than closing the loop on the *observed* violation rate at deployment. CLASP instead is a post-hoc wrapper around any base's scores, enforces cost and latency jointly as violation-rate targets, and adapts online. These routing-layer methods are orthogonal to serving-systems work that meets SLOs at the scheduling layer through continuous batching, disaggregation, and queue management (Yu et al., 2022; Kwon et al., 2023; Agrawal et al., 2024; Zhong et al., 2024; Patke et al., 2024); CLASP acts above the scheduler, at action selection.

3. Problem Setup

Actions. We route among K=33 multi-agent configurations: five topologies (SAS, DEC, IND, CEN, HYB) crossed with GPT-4.1 model tiers (OpenAI, 2025) (T1=nano, T2=mini, T3=large), as detailed in Appendix A (Table 4). For every query i and action j an offline oracle cache records quality Y^q_{ij} (token-level F1), cost c_{ij}, and latency ℓ_{ij}. Each query is represented by a frozen 384-d MiniLM-L6-v2 embedding $\mathbf{x}_i$ (Reimers & Gurevych, 2019; Wang et al., 2020).

Base router. A *base router* is any function that maps a query to a per-action quality score, $g : \mathbf{x}_i \mapsto \mathbf{s}_i \in \mathbb{R}^K$, and routes by $\arg\max_j s_{ij}$. This is all CLASP requires of a base; the score may come from a latent-factor model, a similarity kernel, a classifier, or a topology controller (Section 5).

SLO and the constraint. An SLO is a triple $(\gamma, \tau, \varepsilon)$: a per-query cost budget γ, latency deadline τ, and a tolerated *violation rate* ε. A policy is compliant if both its cost-violation rate $\mathrm{CV} = \Pr[c > \gamma]$ and latency-violation rate $\mathrm{LV} = \Pr[\ell > \tau]$ are at most ε. These are *per-dimension* chance constraints, one on each resource (so the probability of violating *either* is bounded by 2ε via a union bound), not a soft penalty or a single budget. Unlike prior routers that treat cost as an average budget or a soft penalty, CLASP elevates cost to a co-equal probabilistic constraint alongside latency. SLOs shift at deployment; the goal is to keep any base router compliant across regimes without retraining it.

4. Method

CLASP is a wrapper around an already-trained router. It never changes the router's parameters and it never observes the test-time quality of actions it did not choose. For each query, the base router supplies only a vector of *predicted quality scores* $\mathbf{s}_i = g(\mathbf{x}_i)$. CLASP turns these into a *resource-adjusted decision score* by subtracting cost and latency penalties, then chooses the best remaining action.

We keep three quantities separate throughout the paper. Y^q_{ij} is the cached *actual quality* (token-F1) of action j on training/evaluation query i. s_{ij} is the base router's *predicted quality score*; different routers may put this score on very different scales. The score used by CLASP for selection is neither of those: it is s_{ij} after standardization, masking, and resource penalties.

4.1. Offline preparation: learn reusable quantities

Base scores. The base router g is trained or fitted exactly as its original method prescribes, using the cached actual qualities Y^q_{ij} as supervision when needed. After this point CLASP treats g as a black box: for a new query $\mathbf{x}_i$, it asks only for the score vector $\mathbf{s}_i$. Because the five bases emit scores on different numerical scales, CLASP computes a mean μ_g and standard deviation σ_g from the base's training scores and uses the standardized score $\tilde{s}_{ij} = (s_{ij} - \mu_g)/\sigma_g$ in the decision rule below. This keeps the cost and latency multipliers comparable across heterogeneous routers.

Resource predictor. The base score says which action looks high-quality; it does not say whether that action will be cheap or fast. CLASP therefore trains its own small MLP $h_\phi : \mathbf{x}_i \mapsto (\hat{\mathbf{c}}_i, \hat{\boldsymbol{\ell}}_i) \in \mathbb{R}^K \times \mathbb{R}^K$ to predict the cost and latency of every action for the query. It is trained once on the cached cost and latency values (normalized by their 95th percentiles for scale stability), is independent of the base router, and predicts raw resources rather than a particular SLO's violations.

4.2. SLO binding: instantiate the controller

When the operator supplies an SLO $(\gamma, \tau, \varepsilon)$, CLASP binds the reusable offline quantities to that regime. This is lightweight: the base router and h_ϕ are not retrained.

Feasibility mask. First, CLASP removes actions that are chronically unsafe under the target SLO. For each action j, it computes the training-set violation rates $\hat{v}^c_j = \frac{1}{N}\sum_i \mathbb{1}[c_{ij} > \gamma]$ and $\hat{v}^\ell_j$ (analogously for latency), then keeps only

$$\mathcal{F} = \{\, j : \hat{v}^c_j \le \kappa\varepsilon \;\wedge\; \hat{v}^\ell_j \le \kappa\varepsilon \,\},$$

where κ is a feasibility-strength knob (default κ=2). Intuitively, an action that violates the budget or deadline too often on the training cache should not be a deployment candidate for this SLO.

Penalty weights. Second, CLASP initializes two penalty weights, λ_c for cost and λ_ℓ for latency. These weights answer a simple question: how strongly should the controller avoid predicted cost or latency overshoot? We calibrate them by replaying the selection-and-update rule below on a held-out calibration split, starting from λ=0. For the main static experiments, the calibrated weights are then frozen before test evaluation, so no test outcome influences the reported results. When the operator changes the SLO, CLASP re-binds by recomputing only $\mathcal{F}$ and the two λ values (the base router and h_ϕ are untouched). This discrete re-binding is distinct from the online loop of Section 4.3, which adapts to distribution *drift* under a *fixed* SLO; an SLO change is an explicit operator action, not something CLASP auto-detects.

4.3. Deployment: adjust quality scores and choose

At deployment, a query flows through the base router and CLASP in sequence. The base router proposes quality scores; CLASP removes infeasible actions and subtracts predicted resource penalties. For a query with standardized base scores $\tilde{s}_{ij}$ and predicted resources $(\hat{\mathbf{c}}_i, \hat{\boldsymbol{\ell}}_i)$, CLASP selects

$$a^* = \arg\max_{j \in \mathcal{F}} \; \tilde{s}_{ij} \; - \; \lambda_c P^c_{ij} \; - \; \lambda_\ell P^\ell_{ij}, \tag{1}$$

with query-adaptive penalties $P^c_{ij} = \max(0, \hat{c}_{ij}/\gamma - 1)$ and $P^\ell_{ij} = \max(0, \hat{\ell}_{ij}/\tau - 1)$, each zero inside the budget/deadline and growing linearly with the predicted overshoot. Thus a high-quality action can still win, but it must be high-quality enough to justify any predicted cost or latency risk. This is the Lagrangian relaxation of "maximize quality subject to the cost and latency constraints."

Online adaptation: a closed loop. After serving a^*, CLASP observes the realized cost c and latency ℓ and feeds them back into the controller, closing the loop with two coupled updates. First, it nudges each penalty weight toward the target violation rate,

$$\lambda_k \leftarrow \mathrm{clip}\big[\lambda_k + \eta\,(v^k_t - \varepsilon),\, 0,\, \lambda_{\max}\big], \tag{2}$$

where $v^c_t = \mathbb{1}[c > \gamma]$ and $v^\ell_t = \mathbb{1}[\ell > \tau] \in \{0, 1\}$: if violations exceed ε the weight rises and decisions become more conservative, if they fall below it relaxes, so it tends toward the target rate ε (a heuristic dual-ascent update, not a convergence guarantee; the clip keeps weights non-negative and prevents windup). This is the same rule that calibrates λ offline (Section 4.2). Second, it refreshes its per-action resource estimates, which matters when deployment *drifts*: the offline predictor h_ϕ goes stale (e.g. a load spike slows, or a price change raises the cost of, some backends), so CLASP keeps a running estimate, initialized from the offline mean and updated by an exponential moving average,

$$\tilde{r}_j \leftarrow (1-\alpha)\,\tilde{r}_j + \alpha\, r^{\mathrm{obs}}, \quad r \in \{c, \ell\}, \tag{3}$$

after which the penalties in Eq. 1 use $\tilde{c}_j, \tilde{\ell}_j$. Under a stable deployment the weight update alone holds the violation rate at ε; under drift the two complement each other, as the estimate refresh tells the controller which actions have gone bad while the weight ascent makes the penalty strong enough to avoid them (Section 6.5). Throughout, the base router and predictor are untouched.

Algorithm 1 CLASP (deploy under SLO $(\gamma, \tau, \varepsilon)$). In the *static* experiments (Tables 1–3) the weights λ are fixed after offline calibration; the per-query λ and estimate updates (line 5) are exercised in the online/drift setting (Section 6.5).

Offline: base g, predictor h_ϕ, standardization μ_g, σ_g, cache rates $\hat{v}^c, \hat{v}^\ell$

1: $\mathcal{F} \leftarrow \{j : \hat{v}^c_j \le \kappa\varepsilon \wedge \hat{v}^\ell_j \le \kappa\varepsilon\}$; calibrate λ_c, λ_ℓ by Eq. 2
2: **for** each deployed query $\mathbf{x}_i$ **do**
3: $\tilde{\mathbf{s}}_i \leftarrow (g(\mathbf{x}_i) - \mu_g)/\sigma_g$; $(\hat{\mathbf{c}}_i, \hat{\boldsymbol{\ell}}_i) \leftarrow h_\phi(\mathbf{x}_i)$
4: select a^* by Eq. 1 over $\mathcal{F}$; serve it
5: observe c, ℓ; update $\tilde{c}, \tilde{\ell}$ by Eq. 3 and λ by Eq. 2
6: **end for**

5. Experimental Setup

Data. Five datasets span multi-hop QA (MuSiQue (Trivedi et al., 2022)), competition math (MATH L3–5 (Hendrycks et al., 2021)), and domain knowledge (MMLU-Pro Law, Chemistry, Physics (Wang et al., 2024)), all public research benchmarks, balanced to 300 training and 99 test queries per domain (1,500 training and 495 test total) by subsampling a larger shipped pool, with the surplus concentrated in MuSiQue. All 33 actions are pre-evaluated per query (F1, cost, latency) in an offline cache, so all experiments run without any new LLM calls: once the cache is built, every result in this paper runs on a single CPU in minutes.

Base routers. We reimplement five published routers, spanning five mechanisms, on the shared 33-action space (Appendix B): **IRT-Router** (Song et al., 2025) (latent-factor, $\sigma(\mathbf{a}_i^\top \boldsymbol{\theta}_j + b_i)$); **EmbedLLM** (Zhuang et al., 2025) (matrix factorization, $p_j \odot$ projected query); **RouterDC** (Chen et al., 2024) (dual-contrastive embedding, cosine similarity); **RouteLLM-SW** (Ong et al., 2024) (similarity-weighted kNN over training outcomes); and **MasRouter** (Yue et al., 2025) (VAE dual-tower topology controller). Each emits per-action quality scores; CLASP is then attached unchanged.

SLO levels. We use three regimes with ε=5% (a standard P95 service level (Beyer et al., 2016)), set relative to the action-space cost/latency distribution rather than arbitrarily. The cost budget is anchored to the mean cost c^*=\$0.016 of the highest-quality action (IND-T2-T3): **tight**, **moderate**, and **relaxed** use $\gamma \approx 1.5, 1.9, 2.5\, c^*$ (\$0.025, \$0.030, \$0.040), spanning budgets from tight headroom over the best action to generous. The latency deadline is anchored at percentiles of the per-action latency distribution: tight/moderate/relaxed use τ=50, 85, 125 s (the P50/P75/P90 of query–action pairs), loosening monotonically like the cost budget. Following the tail-percentile convention for latency SLOs in serving (Agrawal et al., 2024; Zhong et al., 2024), moderate and relaxed are tail anchors; the tight deadline is the median, deliberately *binding by design* (about half the menu is too slow) so that routing is non-trivial. This median is an anchor over the action *menu*, not a served-traffic target: after routing, the policy must still satisfy the 95% chance constraint. These levels induce a routing-necessity gradient, and Section 6.4 sweeps the regime continuously to confirm the findings are not specific to these anchors.

Metrics and protocol. We report token-F1 (quality) and cost/latency violation rates CV/LV (%). The dual variables λ are calibrated, and the predictor's early stopping decided, on a validation split carved from training; κ and η are fixed defaults; the test set is touched once. Results are 5-seed means; non-learned bases are deterministic. With 495 test queries, a violation rate near ε carries a binomial standard error of about 1 pp (about 2 pp on a single 99-query domain), so cells within $\sim$1 pp of the 5% line should be read as *at* the boundary rather than decisively compliant or not. Where a single regime is shown (the headline F1, the component ablation, and the drift study), we use the *moderate* SLO as a representative midpoint; the continuous SLO sweep (Section 6.4) confirms the conclusions hold across regimes, not just at this anchor.

6. Results

6.1. CLASP retrofits SLO compliance onto every base

Table 1 reports each base router alone (route by quality) versus with CLASP attached. Run alone, most bases violate the SLOs heavily, with cost/latency violation rates as high as 51% (far above the 5% target), because they optimize quality and ignore the constraints; the two strongest bases (RouterDC, RouteLLM-SW) already sit near compliance under loose SLOs but still breach them as the SLO tightens. With CLASP attached and *no base retraining*, violations collapse (Figure 1): all five bases are compliant at *both* the moderate and relaxed SLOs, and only the tightest SLO leaves residuals (two of five remain compliant),

Table 1. CLASP retrofits cost/latency SLO compliance onto five mechanism-diverse routers (5-seed mean). For each base we show route-by-quality (*alone*) vs. +*CLASP*. CV/LV are cost/latency violation rates (%); a regime is compliant when both CV and LV are $\leq \varepsilon$=5%. F1 is quality (regime-independent for *alone*; reported at the moderate SLO for +*CLASP*).

			TIGHT		MOD.		RELAX.	
Base		F1	CV	LV	CV	LV	CV	LV
IRT-Router	alone	0.770	22.3	27.5	18.9	17.9	14.9	7.1
	+CLASP	0.799	4.0	4.9	2.2	1.0	1.2	0.3
EmbedLLM	alone	0.651	25.1	51.2	21.8	25.9	16.4	7.9
	+CLASP	0.775	3.0	3.9	1.6	0.7	0.6	0.7
RouterDC	alone	0.788	4.6	8.0	3.7	2.7	2.7	0.8
	+CLASP	0.795	3.2	5.8	1.8	1.1	0.9	0.2
RouteLLM-SW	alone	0.817	15.4	5.9	6.1	1.4	3.2	0.4
	+CLASP	0.816	7.2	4.4	4.1	1.0	3.2	0.4
MasRouter	alone	0.805	23.7	16.6	17.8	9.9	15.2	4.6
	+CLASP	0.791	4.3	5.04	2.3	1.1	2.7	0.2

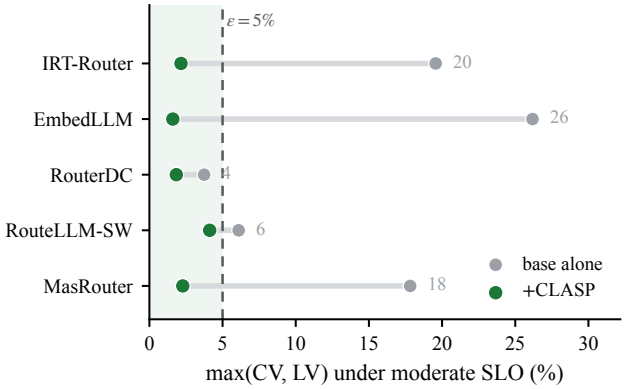

Figure 1. Base routers alone violate the moderate SLO by up to $\sim$26%; with CLASP attached (no base retraining) every one reaches the 5% target. 5-seed means.

where violations are still cut sharply, by up to $\sim$13$\times$ for the heaviest violators. Quality is improved for the weaker bases (CLASP's feasibility filter raises EmbedLLM from 0.65 to 0.78 by removing actions that are both expensive and low-quality) and changes the strong bases by at most 1.4 pp across regimes (at moderate, RouteLLM-SW 0.817$\rightarrow$0.816, MasRouter 0.805$\rightarrow$0.791).

The preserved quality leaves headroom, as expected for text-feature routing: a full-information *constrained oracle* (per query, the highest-F1 action satisfying both constraints) attains F1 0.90–0.92, so CLASP ($\approx$ 0.79) is compliant and competitive but not near-optimal. Across seeds the standard error of the mean F1 is $\leq$ 0.01 for the learned bases; RouteLLM-SW's base scores are deterministic given the cache (its +CLASP spread comes only from the controller's calibration seed). The residuals at the tight SLO are informative: RouteLLM-SW (expensive-action-biased) leaves a cost

residual and RouterDC/MasRouter (slow-action-biased) a latency residual; these residuals are the tight-regime face of the compliance–quality frontier (Section 6.4).

6.2. Comparison to non-oracle static baselines

The constrained oracle bounds CLASP from above; we also anchor it from below against two non-oracle constrained baselines a practitioner could deploy without it, both run on the same cache (Table 2). MEANFILTER keeps only actions whose per-action training-mean cost and latency meet the budget and deadline, then routes by quality; STATICLAG routes by quality minus fixed penalties on each action's mean cost and latency overshoot, the two weights grid-searched on the training split to hit ε. Both cut violations far below the unconstrained base, so they are non-trivial. But because they constrain each action's *mean* rather than its ε-quantile, they stay non-compliant on the two bases whose preferred actions are mean-feasible yet tail-risky (RouteLLM-SW 6.1% and MasRouter 5.7–5.9%), exactly the bases where CLASP's violation-rate filter and query-adaptive penalty reach compliance (4.1%, 2.3%). And on the weak EmbedLLM base, where both static baselines do comply, they reach only 0.67 while CLASP's filter lifts it to 0.78 ($+11$ pp over them, $+13$ over the 0.65 base). CLASP is the only one of the four compliant on all five bases at the moderate and relaxed SLOs (on the mean), staying within 1.4 pp of each base's quality and improving the weak ones. The seed std sharpens this: MasRouter's base routing swings wildly across seeds (viol 17.8 ± 17.3) and drags the static baselines' near-ε numbers with it (5.7–$5.9\pm\sim3.8$), so their non-compliance there is not seed-robust; CLASP's seed std stays ≤0.9 on every base, making its compliance a stable property rather than a seed-lucky mean. Table 2 reports all three SLOs; the baselines' shortfall only widens as the SLO tightens (MasRouter MeanFilter/StaticLag reach 12.1% at the tight SLO, versus CLASP 5.0%).

6.3. What each component contributes

Table 3 isolates CLASP's parts across all three SLOs. The *feasibility filter* does the heavy lifting: filter-only already brings every base into compliance at the moderate and relaxed SLOs and delivers the quality gains (EmbedLLM $0.65 \rightarrow 0.78$) by discarding actions that are both costly and weak. The *dual ascent* alone is insufficient for *static* compliance, leaving residual violations for some bases (EmbedLLM LV 7.0, RouteLLM-SW CV 5.6): its penalty depends on predicted latency, which is hard to read from query text (our latency predictor reaches only $R^2{=}0.10$, versus 0.68 for cost). Under *static* (offline) conditions, then, the filter is the workhorse and the online closed loop adds little. It earns its keep only under *drift*, where the full loop restores compliance that the offline-fixed filter cannot, and on the hardest-hit bases both of its moves are usually needed (ablating either half typically stays in violation; Section 6.5). Dual-only's shortfall is largest at the tight SLO (Table 3).

Table 2. Static baselines vs. CLASP at all three SLOs (tight/moderate/relaxed; 5-seed mean). For each base and SLO we show F1 and the violation rate *viol* (%, larger of cost and latency) for Base, MeanFilter, StaticLag, and CLASP; compliant when viol$\leq \varepsilon{=}5\%$. Discussion, including across-seed stability, in Section 6.2.

		Base		MeanFilter		StaticLag		CLASP	
Base		F1	viol	F1	viol	F1	viol	F1	viol
IRT-Router	T	0.77	27.5	0.80	7.0	0.80	6.9	0.79	4.9
	M	0.77	18.9	0.78	2.8	0.78	3.6	0.80	2.2
	R	0.77	14.9	0.77	1.9	0.77	4.1	0.78	1.2
EmbedLLM	T	0.65	51.2	0.78	5.1	0.78	5.2	0.77	3.9
	M	0.65	25.9	0.67	4.2	0.67	4.3	0.78	1.6
	R	0.65	16.4	0.65	3.4	0.65	3.4	0.67	0.7
RouterDC	T	0.79	8.0	0.80	5.7	0.80	5.7	0.79	5.8
	M	0.79	3.7	0.79	1.9	0.79	2.2	0.80	1.8
	R	0.79	2.7	0.79	1.0	0.79	1.1	0.79	0.9
RouteLLM-SW	T	0.82	15.4	0.82	15.4	0.82	15.4	0.81	7.2
	M	0.82	6.1	0.82	6.1	0.82	6.1	0.82	4.1
	R	0.82	3.2	0.82	3.2	0.82	3.2	0.82	3.2
MasRouter	T	0.80	23.7	0.80	12.1	0.80	12.1	0.80	5.0
	M	0.80	17.8	0.80	5.7	0.80	5.9	0.79	2.3
	R	0.80	15.2	0.80	2.7	0.80	3.0	0.80	2.7

Table 3. Component ablation at all three SLOs (tight/moderate/relaxed; 5-seed mean; F1 / CV / LV per cell). *alone* routes by quality; then *filter-only*, *dual-only*, and *full*. Discussion in Section 6.3.

Base		alone	filter-only	dual-only	full
IRT-Router	T	0.77/22.3/27.5	0.79/4.3/4.9	0.80/5.3/6.7	0.79/4.0/4.9
	M	0.77/18.9/17.9	0.80/2.2/1.0	0.78/4.2/3.8	0.80/2.2/1.0
	R	0.77/14.9/7.1	0.78/1.2/0.3	0.77/4.6/2.6	0.78/1.2/0.3
EmbedLLM	T	0.65/25.1/51.2	0.77/2.9/3.9	0.75/3.8/9.9	0.77/3.0/3.9
	M	0.65/21.8/25.9	0.78/1.6/0.7	0.67/3.9/7.0	0.78/1.6/0.7
	R	0.65/16.4/7.9	0.67/0.6/0.7	0.65/5.0/3.9	0.67/0.6/0.7
RouterDC	T	0.79/4.6/8.0	0.79/3.2/5.7	0.79/2.8/5.7	0.79/3.2/5.8
	M	0.79/3.7/2.7	0.80/1.8/1.1	0.79/2.3/1.4	0.80/1.8/1.1
	R	0.79/2.7/0.8	0.80/0.9/0.2	0.79/1.8/0.6	0.80/0.9/0.2
RouteLLM-SW	T	0.82/15.4/5.9	0.81/8.3/4.8	0.81/11.3/5.1	0.81/7.2/4.4
	M	0.82/6.1/1.4	0.81/4.4/1.0	0.82/5.6/1.3	0.82/4.1/1.0
	R	0.82/3.2/0.4	0.82/3.2/0.4	0.82/3.2/0.4	0.82/3.2/0.4
MasRouter	T	0.81/23.7/16.6	0.80/5.1/5.0	0.80/8.1/6.2	0.80/4.3/5.0
	M	0.81/17.8/9.9	0.79/2.3/1.1	0.80/4.8/1.2	0.79/2.3/1.1
	R	0.81/15.2/4.6	0.80/2.7/0.2	0.80/3.0/0.4	0.80/2.7/0.2

6.4. The compliance–quality frontier

Feasibility strength. Figure 2 sweeps the filter knob κ at the tight SLO, the one regime where compliance is non-trivial to obtain (at looser SLOs it costs essentially no quality; Figure 3), so the quality cost of stricter filtering is visible there. Tightening to $\kappa{=}1$ ($\hat{v}_j \leq \varepsilon$) makes *all* bases fully compliant (CV$\rightarrow 0$, LV well below ε at $\sim$2–3%), but collapses tight-SLO F1 to ≈ 0.70 (below the best single action feasible at the default filter, IND-T2-T3 at F1 0.81), be-

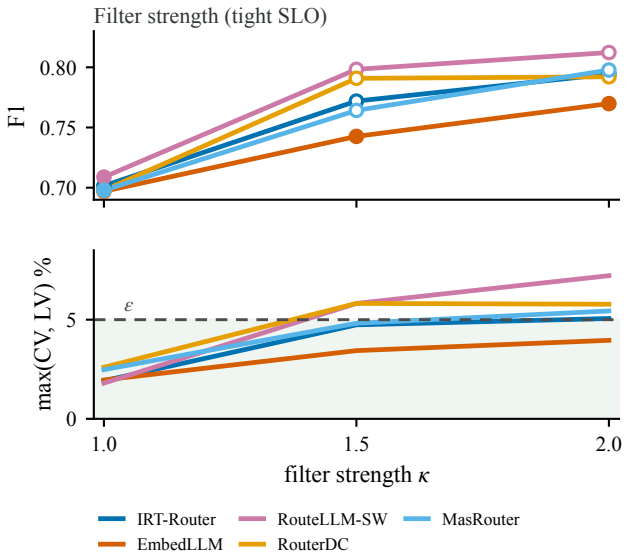

Figure 2. Feasibility strength κ at the tight SLO (5-seed mean). Stricter filtering (smaller κ) drives the violation rate below ε (bottom) but drops F1 (top); looser κ keeps F1 high but leaves residual violations (e.g. RouteLLM-SW). Filled markers denote compliant regimes.

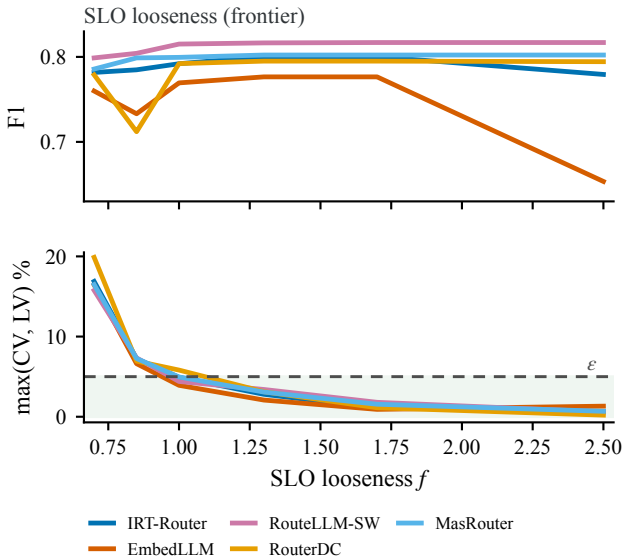

Figure 3. SLO looseness f: scaling the SLO $(\gamma, \tau) = f \cdot (0.03, 50)$ traces the compliance–quality frontier continuously (5-seed mean; base colors as in Figure 2). Every base enters the compliant region as the SLO loosens while quality stays high (Section 6.4).

cause almost only ultra-cheap, low-quality actions survive the mask. The lenient $\kappa{=}2$ keeps F1 high but leaves the residuals above. Thus compliance at the tight SLO is not free: it must be traded against quality, and κ is the dial.

SLO tightness. Scaling the SLO as $(\gamma, \tau) = f \cdot (0.03, 50)$ traces the frontier continuously (Figure 3). Compliance is monotone in looseness f: at $f{=}0.7$ every base violates (latency $\sim$16–20%), by $f{\geq}1.3$ all comply, and each base has a compliance threshold, with latency-biased RouterDC needing the loosest SLO. The mechanism is the feasibility filter: the number of actions passing it widens from 8 of 33 at the tight SLO to 11 at moderate and 18 at relaxed, and the high-quality IND configurations enter as it loosens, so compliance costs essentially no quality once the best actions are admitted (the remaining masked actions are dominated, being both costlier and lower-quality). At very loose SLOs the controller goes inactive and F1 returns to the base's raw quality (EmbedLLM falls to 0.65), so CLASP's quality benefit peaks at intermediate tightness, where the filter discards genuinely bad actions. The three hand-picked regimes are thus representative points on a smooth frontier, not artifacts.

6.5. Online adaptation to deployment drift

The filter and predictor assume the cache reflects deployment; when the deployed cost/latency *drifts*, that assumption breaks and a static controller fails. We stream test queries through CLASP on the IRT-Router base at the moderate SLO and, midway, trigger a drift on the router's three preferred (highest-quality feasible) actions, along each resource in turn: a *latency* spike makes them $3.5\times$ slower, and a *cost* spike makes them $3.5\times$ more expensive, while the offline filter and predictor are unchanged (Figure 4, top and bottom). Adapting only *one* half of the loop fails to recover in either case (Figure 4): with the estimates left stale the weights climb but penalize the wrong, offline-predicted actions, and with the weights frozen the refreshed estimates are penalized too weakly, so the windowed violation rate stays high both ways (38% under latency drift, 54% under cost). Only the *full* closed loop, refreshing the per-action estimates while the weights ascend, retargets the drifted actions and returns the rate to ε: violations recover to 2% under both. The gap is robust to the spike magnitude: across 3–5$\times$ spikes on 3–5 actions the full loop holds at 0–6% while static degrades to 30–80%; only a small $2\times$ spike, which barely moves an action's mean across the deadline, escapes the per-action-mean detector, and the full loop then recovers only partially (8–16%). It also generalizes beyond IRT-Router (the full per-base decomposition is Table 5, Appendix D): across all five bases the full loop returns the windowed rate to 0–8% under both drifts (within ε but for two residual endpoints at 6–8%), while a single half alone leaves 14–84% violations on the bases the perturbation hits hardest (IRT-Router, EmbedLLM, RouteLLM-SW). Both updates are needed on most of these; the one exception is RouteLLM-SW under cost drift, where the estimate refresh alone already recovers (4%). On the remaining bases the $3.5\times$ shift barely registers. This is the regime where the on-

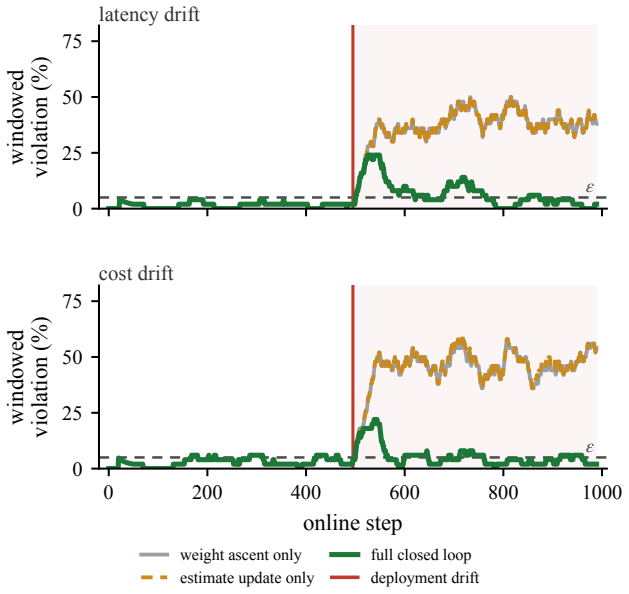

Figure 4. Online adaptation under deployment drift (IRT-Router base). At the marked step a drift hits the router's preferred actions ($3.5\times$): a latency spike (top) and a cost spike (bottom), unseen by the offline filter and predictor. Adapting only the weights (ascent on stale estimates) or only the estimates (EMA with fixed weights) stays in violation (38–54%); only the full online closed loop recovers compliance ($\leq \varepsilon$).

line machinery earns its keep: under no drift the offline filter alone suffices (Section 6.3), but when deployment departs from the offline cache, online adaptation is what restores compliance. A complementary, easier axis is an operator *SLO change*, where CLASP simply recomputes the mask and recalibrates λ, again with no base retraining.

7. Discussion

Why post-hoc and modular. The experiments show that a small, detachable controller (feasibility filter, query-adaptive cost/latency predictors, and an online violation-rate closed loop) suffices to make heterogeneous quality-only routers satisfy joint cost/latency SLOs without retraining them. This is complementary to constraint-in-training routers (Bhatti et al., 2026; Wang et al., 2025): those build one constrained policy, whereas CLASP turns an entire ecosystem of existing routers SLO-aware and tracks shifting regimes by recalibrating two scalars.

The frontier is the message. A naive reading of Table 1 is "the tool sometimes leaves residuals"; the sweeps show the residuals are the tight-regime face of a genuine compliance–quality frontier. Full compliance is always attainable (Figure 2, κ=1) but, at tight SLOs, only by sacrificing quality, because the feasible high-quality actions truly violate. CLASP's contribution is to make this frontier explicit and operator-tunable rather than implicit in a fixed policy.

Limitations. Aggregate compliance can mask per-subgroup violations: across our five domains, at the moderate SLO IRT-Router+CLASP is compliant overall yet still exceeds the latency target on MATH (LV 6%) and the cost target on Chemistry (CV 8%) and Physics (CV 4%), so operators should track per-domain SLOs. CLASP targets the chance constraint heuristically and offers no formal feasibility guarantee, so the tight-SLO residuals above ε in Table 1 are expected rather than anomalous. Our deployment-drift study uses *controlled synthetic* perturbations of the cache (multiplicative load spikes), not real production traces, which we cannot collect here; the per-action running estimate also tracks only shifts large enough to move an action's mean across the deadline (a $2\times$ spike is below this threshold, and a drift that fattens an action's tail while leaving its mean feasible is likewise invisible to a mean-based estimator), and our online study spans five bases over one action space. Real-trace drift, a second action space, and finer-grained (e.g. query-conditioned) online estimators are future work.

8. Conclusion

We presented CLASP, a post-hoc controller that retrofits joint cost–latency SLO compliance onto any per-action-scoring LLM router without retraining it, and adapts across shifting regimes by recalibrating two dual variables. Across five mechanism-diverse published routers, CLASP cuts large SLO violations to the target while preserving quality, and turns the underlying compliance–quality frontier into an explicit knob that operators can tune per regime. By decoupling constraint control from the router, CLASP offers a reusable path to SLO-aware deployment of the many quality-only routers already in use.

Impact Statement

This work helps deploy LLM routers under cost and latency SLOs without retraining, potentially reducing the compute and energy of serving multi-agent systems. Cost-aware routing can concentrate load on cheaper, weaker configurations and degrade quality for some query types; CLASP keeps an explicit quality term and the compliance–quality frontier makes this trade-off visible, but practitioners should monitor per-subgroup quality. Experiments use public benchmarks and cached outputs; no personal data is used.

References

Agrawal, A., Kedia, N., Panwar, A., Mohan, J., Kwatra, N., Gulavani, B. S., Tumanov, A., and Ramjee, R. Taming throughput-latency tradeoff in LLM inference with

Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

Beyer, B., Jones, C., Petoff, J., and Murphy, N. R. *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, 2016.

Bhatti, A. S., Vaddina, V., and Birru, D. PROTEUS: SLA-aware routing via lagrangian RL for multi-LLM serving systems. *arXiv preprint arXiv:2601.19402*, 2026.

Chen, L., Zaharia, M., and Zou, J. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Chen, S., Jiang, W., Lin, B., Kwok, J. T., and Zhang, Y. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Ruhle, V., Lakshmanan, L. V., and Awadallah, A. H. Hybrid LLM: Cost-efficient and quality-aware query routing. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Hu, Q. J., Bieker, J., Li, X., Jiang, N., Keigwin, B., Ranganath, G., Keutzer, K., and Upadhyay, S. K. RouterBench: A benchmark for multi-LLM routing system. *arXiv preprint arXiv:2403.12031*, 2024.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

Mei, K., Xu, W., Guo, S., Lin, Z., and Zhang, Y. OmniRouter: Budget and performance controllable multi-LLM routing. *arXiv preprint arXiv:2502.20576*, 2025.

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. RouteLLM: Learning to route LLMs with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

OpenAI. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/, April 2025. Accessed: 2026-05-05.

Patke, A., Reddy, D., Jha, S., Qiu, H., Pinto, C., Narayanaswami, C., Kalbarczyk, Z., and Iyer, R. Queue management for SLO-oriented large language model serving. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC)*, 2024.

Reimers, N. and Gurevych, I. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Song, W., Huang, Z., Cheng, C., Gao, W., Xu, B., Zhao, G., Wang, F., and Wu, R. IRT-router: Effective and interpretable multi-LLM routing via item response theory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Su, J., Xia, Y., Lan, Q., Song, X., Chen, C., Yang, J., He, L., and Shi, T. Difficulty-aware agentic orchestration for query-specific multi-agent workflows. *arXiv preprint arXiv:2509.11079*, 2025.

Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. MuSiQue: Multihop questions via single hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33: 5776–5788, 2020.

Wang, W., Yang, T., Chen, H., Zhao, Y., Dernoncourt, F., Rossi, R. A., and Eldardiry, H. Learning to route LLMs from bandit feedback: One policy, many trade-offs. *arXiv preprint arXiv:2510.07429*, 2025.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., Li, T., Ku, M., Wang, K., Zhuang, A., Fan, R., Yue, X., and Chen, W. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024.

Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., and Chun, B.-G. Orca: A distributed serving system for transformer-based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2022.

Yue, Y., Zhang, G., Liu, B., Wan, G., Wang, K., Cheng, D., and Qi, Y. MasRouter: Learning to route LLMs for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Zhang, G., Yue, Y., Sun, X., et al. G-Designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*, 2024.

Zhang, G., Niu, L., Fang, J., Wang, K., Bai, L., and Wang, X. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*, 2025.

Zhong, Y., Liu, S., Chen, J., Hu, J., Zhu, Y., Liu, X., Jin, X., and Zhang, H. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

Zhuang, R., Wu, T., Wen, Z., Li, A., Jiao, J., and Gonzalez, J. E. EmbedLLM: Learning compact representations of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

A. Topology Summary

Table 4. Action-space construction from topology and model-tier assignments. Tiers are T1=nano, T2=mini, and T3=large.

Topology	Roles	Tier binding	Actions
SAS	self-refining single agent	one shared tier	3
DEC	three debating peers	one shared peer tier	3
IND	independent workers + synthesizer	brain tier × worker tier	9
CEN	orchestrator + workers	brain tier × worker tier	9
HYB	orchestrator + workers + peer review	brain tier × worker tier	9
Total			33

B. Base-Router Reimplementations

All bases are reimplemented from the authors' released code on the shared 33-action space with 384-d MiniLM query features and full-information per-action token-F1 labels; cost and latency are handled solely by CLASP.

IRT-Router (Song et al., 2025). A learnable per-action ability vector $\boldsymbol{\theta}_j$ (from an action embedding) and a query-conditioned discrimination $\mathbf{a}_i = \mathrm{softplus}(W\mathbf{x}_i)$ and difficulty b_i, with $s_{ij} = \sigma(\mathbf{a}_i^\top \boldsymbol{\theta}_j + b_i)$ trained by binary cross-entropy on F1 (latent dim 25).

EmbedLLM (Zhuang et al., 2025). Text-conditioned matrix factorization: learnable action embeddings $\mathbf{p}_j$ and a projected query $W\mathbf{x}_i$, interaction $\mathbf{p}_j \odot W\mathbf{x}_i$ read out by a linear head to a per-action quality, trained with a continuous BCE on F1 (latent dim 64, train-time query noise).

RouterDC (Chen et al., 2024). A query MLP and 33 learnable action embeddings scored by cosine similarity, trained with the sample-LLM dual contrastive (InfoNCE) loss (top-k positives / bottom-k negatives by F1) plus a k-means cluster contrastive loss.

RouteLLM-SW (Ong et al., 2024). Non-parametric similarity-weighted scoring: per-action quality is the training F1 averaged over training queries with the original cosine-similarity weighting $10 \cdot 10^{\mathrm{sim}/\max \mathrm{sim}}$.

MasRouter (Yue et al., 2025). A VAE dual-tower: query and action towers encode to L2-normalized latents, scored by temperature-softmax cosine similarity; trained with a listwise loss to the F1 distribution plus the VAE KL term (the original REINFORCE/MAS rollout is replaced by the full-information F1 supervision).

C. CLASP Configuration

CLASP's cost/latency predictor is a two-layer MLP (128→64, dropout 0.3) trained with MSE on 95th-percentile-normalized cost and latency. We fix the dual step η=0.1; the controller is insensitive to it (across $\eta \in \{0.01, \ldots, 10\}$, three orders of magnitude, F1 and both violation rates shift by $\leq$0.7 pp at every base and SLO), because the filter carries most of the compliance and the dual only fine-tunes. $\lambda_{\max}$=50; default feasibility strength κ=2. Adaptation across SLOs recomputes the mask and recalibrates λ over three passes on the calibration set. For the deployment-drift study, the per-action online estimates use EMA rate α=0.2, and the simulated load spike multiplies the latency of the router's three highest-quality feasible actions by 3.5× from the stream midpoint.

D. Per-Base Drift Recovery

Table 5 gives the full per-base numbers discussed in Section 6.5: for each base, the same 3.5× drift on *its own* three preferred feasible actions at the moderate SLO, along latency and cost in turn, under the two half-loops (weights-only *w*, estimate-only *e*) and the full loop (*f*).

Table 5. Drift recovery across all five bases (moderate SLO; 3.5× on each base's three preferred actions). Windowed violation rate (%) at the end of the stream. *w*: dual-ascent on stale estimates (weights only); *e*: EMA estimate refresh with frozen weights; *f*: full loop. A mode is compliant when $\leq \varepsilon$=5%.

	Latency drift			Cost drift		
Base	w	e	f	w	e	f
IRT-Router	38	38	2	54	54	2
EmbedLLM	14	14	2	30	32	8
RouterDC	0	0	0	2	2	2
RouteLLM-SW	44	44	0	84	4	2
MasRouter	6	6	6	14	6	4