

Handoff Curves: Initial Selection, Later Handoff, and Supervision Targets in Multi-turn LLM Routing

Daniel Matos Lu Liu Yanfeng Zhang*

Riemann Lab, Huawei Technologies

Abstract

We study how initial model selection, later handoff, and training targets affect success and cost in LLM tasks that span multiple turns. We compare several router architectures using handoff curves, which record task success and cost as the handoff step changes. Our dataset contains 34,204 complete executions of three Qwen models across BFCL, MultiWOZ, and τ -retail, providing offline supervision for routing decisions. In our primary comparison, with Qwen3.5 35B-A3B as the starting model, allowing a later handoff does not establish an improvement in utility over initial selection on BFCL or MultiWOZ, and increases cost on BFCL across a range of cost weights. Changing the supervision targets improves router performance on both our dataset and TwinRouterBench. On our dataset, replacing noisy training success curves with monotone fits raises success by 0.2 to 4.1 percentage points at the same cost across the reported operating points. On TwinRouterBench, we construct targets that describe the model capability needed for the remaining work and difficult steps ahead. Compared with training on the benchmark’s per-step labels, these targets raise trajectory pass scores by 7.8 percentage points and cost savings by 6.3 percentage points on the full static bank. A live deployment of the router trained with these targets resolves 79 of 100 instances at a router cost of \$0.59 per instance.

1 Introduction

Large language model (LLM) routing decides which model should serve each query, task, or user, usually trading answer quality against cost. Early LLM routing work used cost-aware cascades (Chen et al., 2023) and routers trained on benchmark outcomes (Shnitzer et al., 2024), and later work added pairwise routers and shared benchmarks (Ding et al., 2024; Ong et al., 2025; Hu et al., 2024).

These routers were designed for a single query, but the field has since moved to multi-turn routing (Zhang et al., 2026c,b; Xu et al., 2026; Yang et al., 2026).

Existing multi-turn routers construct supervision from realised trajectories, verified per-step downgrades, search-derived policies, or annotated handoff windows (Zhang et al., 2026c; Yang et al., 2026; Zhang et al., 2026b; Lei and Huang, 2026). These labels use outcomes observed later in the interaction, but they do not provide repeated outcomes for committing at each alternative handoff step of the same task. The Handoff Tax measures handoffs at selected switch points (Ganz et al., 2026), and the Replay Gap shows that substituting logged outputs does not reproduce a live continuation (Gonuguntla, 2026). Both results motivate direct measurement of the handoff curve.

We collect these curves on BFCL (Patil et al., 2025), τ -retail (Yao et al., 2025), and MultiWOZ (Budzianowski et al., 2018) by committing to a model handoff at each collected handoff step and measuring task success and cost. These measurements let us ask how much of a router’s success and cost trade-off comes from initial selection, and whether allowing a later handoff adds value.

Each task begins on a starting model, which a router can keep or replace. We evaluate routers from each starting model in our pool (Qwen3 8B, Qwen3.5 35B-A3B, and Qwen3.5 397B-A17B), covering both upward and downward handoffs, to a larger or a smaller model, respectively. We compare two policies that use the same fitted router. Initial selection chooses a model at the first step and keeps it. Later handoff makes the same first choice and, if that choice keeps the starting model, can then hand off to another model at a later step. At the primary 35B start on BFCL and MultiWOZ, the direct router’s later-handoff policy adds no established benefit over initial selection. Initial selection by the factored router improves utility over

*Corresponding author: zhangyanfeng8@huawei.com

the training-selected fixed policy at one tested cost weight on BFCL, but the fixed policy has higher utility at $\lambda \in [0.2, 2]$ on BFCL and is not beaten on MultiWOZ. We also propose new supervision targets, that is, what the router learns to predict. On our dataset, we make each training success curve monotone across handoff steps, either increasing or decreasing depending on the type of handoff. On TwinRouterBench (Yang et al., 2026), which has no measured handoff curves, we adapt this idea by replacing each step’s tier label with the strongest tier needed from that step to the end of the instance. On our dataset, these targets raise the direct router’s utility under initial selection at 7 of the 15 cost weights on BFCL and at 4 on MultiWOZ, with no established decrease at any tested weight. They raise the success of both policies by similar amounts, so later handoff still adds no established benefit over initial selection. On TwinRouterBench, they improve the static scores. Our contributions are as follows:

- We build a dataset of 34,204 executions on BFCL, τ -retail, and MultiWOZ, with replicated handoff curves, billed costs, and both successful and failed outcomes, which we will release publicly.
- We investigate whether a later handoff adds value over initial selection, from every starting model and with the same fitted router.
- We show that changing the supervision targets improves routing at several cost weights on our dataset and in static TwinRouterBench evaluation, under an identical training procedure. In the live TwinRouterBench track, a router trained with these targets resolves 79 of 100 instances at a router cost of \$0.59 per instance.

2 Related Work

Routing policies and supervision. Single-query routers and cascades send each query to one model or escalate it to a stronger one (Chen et al., 2023; Shnitzer et al., 2024; Ding et al., 2024; Ong et al., 2025; Aggarwal et al., 2024). Multi-turn routers learn from trajectory returns or searched policies, as in MTRouter (Zhang et al., 2026c), DialRouter (Zhang et al., 2026b), and SeqRoute (Xu et al., 2026). Other methods replace or defer individual steps, as in TRIM (Kapoor et al., 2026), DMR (Li et al., 2026), and ReDAct (Piatrashyn

et al., 2026), or restart a task on a stronger model when it escalates, as in SWE-Router (Son et al., 2026). These methods make other routing decisions. TACIT-Switch learns when to make one permanent upward handoff, from one rollout per model and a teacher-annotated handoff window (Lei and Huang, 2026). We also study one permanent handoff, but we measure its outcome at every collected handoff step of the same task.

Handoff measurement and routing datasets.

RouterBench provides model–query outcomes without a within-session timing axis (Hu et al., 2024). TwinRouterBench verifies local tier substitutions (tiers are model levels from 0, the cheapest, to 3, the strongest) by executing continuations, retaining substitutions only when the episode passes (Yang et al., 2026). The Handoff Tax evaluates seven fixed switch points (Ganz et al., 2026), the switch-drift study examines a final-turn handoff (Khraishi et al., 2026), and the Replay Gap distinguishes live continuations from logged-output substitution (Gonuguntla, 2026). Our dataset measures successful and failed committed handoffs at collected handoff steps, with two executions of each policy on each task, three starting models, and three datasets. It supplies outcomes and billed costs for router supervision.

Supervision and routing decisions. Routing gains depend on the model pool and its shared failures, on task composition, and on label noise (Chen, 2026a; Lee, 2026). The gap to oracle performance depends on how the oracle is defined under stochastic decoding (Chen, 2026b), and deployable routers realise only part of it (Shihab et al., 2026). Accurate outcome predictions also need not produce better decisions (Zhang et al., 2026a). As in that work, we compare supervision targets under an identical training procedure and decision rule. We also compare initial selection with later handoff under the same fitted router.

3 Methodology

3.1 Measuring Handoff Outcomes

For task x , starting model a , and receiving model b , the handoff policy $\pi_k^{a \rightarrow b}$ lets a take the first k steps and b take all remaining steps, with no further handoff. We call step k the handoff step. The handoff curve is the success and cost of this handoff

as functions of k :

$$\begin{aligned} q_{a \rightarrow b}(x, k) &= \mathbb{E}_{\pi_k^{a \rightarrow b}}[Y \mid x], \\ c_{a \rightarrow b}(x, k) &= \mathbb{E}_{\pi_k^{a \rightarrow b}}[C \mid x]. \end{aligned} \quad (1)$$

Here, $Y \in \{0, 1\}$ is the binary task success, and C is the billed cost of the execution. The expectation is over executions of the policy on task x . We execute every policy and do not estimate the curves from logged trajectories. A solo execution of b supplies the $k = 0$ outcome, while staying on a (a solo execution of a) is a separate action. Each cell in the dataset is one policy $\pi_k^{a \rightarrow b}$ on task x , or one solo model, and contains two separately collected end-to-end executions. Each execution runs the whole episode, including the first k steps by a (the prefix). The mean success of the two executions is the cell’s success label, which averages over randomness in model outputs. The executions share the task and collection conditions, and are not assumed to be statistically independent.

3.2 Supervision Targets

A supervision target is the quantity that a router learns to predict. We change it while the learner, features, splits, and decision rule stay fixed, see Figure 1.

On our dataset, raw labels are the measured success probability of each cell. Projected labels make each training success curve monotone across handoff steps by isotonic regression (Barlow et al., 1972): nonincreasing for upward handoffs and non-decreasing for downward handoffs. We expect this direction because the later an upward handoff, the more steps the weaker model takes, and the later a downward handoff, the more steps the stronger model takes. With two executions per cell, the raw curves are noisy, so the projection replaces each curve with the closest curve that has this direction. More executions per cell would also reduce the noise, but each one costs a full episode. We compare raw and projected labels for the direct router on BFCL and MultiWOZ, see Table 4. Only the training curves are projected. Both versions of the router are scored on the measured, unprojected outcomes of the evaluation tasks. The projection is a modelling choice for the training targets. It does not claim that the true success curves are monotone.

TwinRouterBench (Yang et al., 2026) has no measured handoff curves. Its static bank, see Section 4.3, gives each step j a benchmark label

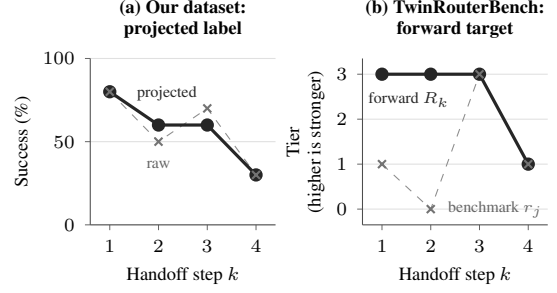


Figure 1: Supervision targets (illustrative values). (a) Raw success of an upward handoff and its projected label. (b) Benchmark labels r_j and forward targets $R_k = \max_{j \geq k} r_j$.

$r_j \in \{0, \dots, 3\}$: the cheapest tier found by the benchmark’s search for which step j still passes, given the tiers already accepted for earlier steps and the strongest tier for all later steps (larger values are stronger models). Along an instance, these labels can rise and fall. We adapt our monotone targets to this setting by transforming the labels rather than fitting them. Each step k receives the forward target

$$R_k = \max_{j=k, \dots, T} r_j, \quad (2)$$

the strongest tier that step k or any later step needs, where T is the final labelled step. If the labels from step k on are 1, 3, 1, then $R_k = 3$, although step k itself needs only tier 1. R_k can only fall as k grows, see Figure 1b. So the indicator $R_k \leq t$, which says whether a permanent handoff to tier t at step k is sufficient under the benchmark labels, can only rise. This is the same monotone direction that our fit gives to the success curve of a downward handoff. We build forward targets from complete training instances and predict them from the observed prefix. These two dataset settings share the monotone direction of the targets but not the method: on our dataset, a fit smooths measured outcomes, whereas on TwinRouterBench, a maximum transforms the original benchmark labels.

3.3 Routers

We study the direct, factored, and burst-aware routers, with SR-KNN as a baseline. All four see only the prefix observed at the decision step and embed text with a frozen MiniLM encoder (Wang et al., 2020; Reimers and Gurevych, 2019). Table 1 lists the data, supervision target, and model of each router. The direct and factored routers run on our dataset and use the decision rule in Section 3.4. The

Router	Data	Supervision target	Model
Direct	Ours	Raw or projected	MLP regressor
Factored	Ours	Projected (timing), raw (difficulty)	Two MLP regressors
Burst-aware	TRB	Benchmark or forward	Boosted-tree classifiers
SR-KNN	Both	Raw (ours), benchmark (TRB)	k -NN ($k = 5$)

Table 1: Routers, the data they run on, what they learn to predict, and their model, see Section 3.2 for the supervision targets. Ours: our dataset. TRB: TwinRouterBench.

burst-aware router runs on TwinRouterBench and makes at most one permanent downward handoff.

Direct router. The direct router regresses the success probability of each cell with a multilayer perceptron with hidden layers of 64 and 32 units and a sigmoid output, trained with cross-entropy against the cell’s success label. It reads the instruction, the observed history, the handoff step, and the starting and receiving models through frozen MiniLM embeddings (Wang et al., 2020; Reimers and Gurevych, 2019), 11 history statistics, such as counts of actions and errors, and model, step, and price features. On BFCL, we train it with ten task-grouped folds and six seeds, and on MultiWOZ and τ -retail, with fixed training and evaluation splits and five seeds, see Appendix A. Unless stated otherwise, results use the direct router trained on raw labels.

Factored router. The factored router splits the same prediction into two parts. The first part, d , predicts how likely the stronger model of the pair is to solve the task on its own. It measures task difficulty and does not depend on timing. The second part, $g(k) = \min\{\tilde{q}(k)/d, 1\}$, predicts how much of that success remains when the handoff happens at step k , and is learned from the isotonic projection $\tilde{q}$ of each training curve, see Section 3.2. The router scores each handoff by the product $d g(k)$ of two regression models of the same kind as the direct router, with g trained on targets between 0 and 1. Unlike the direct router, which can learn any pattern across handoff steps, the factored router separates difficulty from timing.

Burst-aware router. The burst-aware router runs on TwinRouterBench, whose tiers range from 0, the cheapest, to 3, the strongest. At each step, it either retains tier 3 or commits to a cheaper tier $t \in \{0, 1, 2\}$ for the rest of the instance. A burst

is a step labelled $r_k = 3$. Under the benchmark’s scoring rule, a permanent handoff before a later burst causes the instance to fail.

Three gradient-boosted tree classifiers (Friedman, 2001; Ke et al., 2017) read the same features of the observed prefix, see Appendix A, and each contributes one factor to the score. The tier-requirement classifier predicts R_k , the tier needed from step k onward, when trained on forward targets. With benchmark labels, it predicts r_k , the tier needed at step k . In either case, $p_{\text{req}}(t)$ is the predicted probability that the required tier is at most t . The burst classifier gives p_{clear} , the probability that step k is not a burst. With forward targets, it is instead the probability that neither step k nor any of the next three labelled steps is a burst. The local-feasibility classifier is trained only on steps that are not bursts and gives $p_{\text{loc}}(t)$, the probability that $r_k \leq t$ on such a step, which equals 1 for $t = 2$. The router combines the three outputs and calibrates the resulting score:

$$p_t = I_t \left(\sqrt{p_{\text{req}}(t) p_{\text{clear}} p_{\text{loc}}(t)} \right), \quad (3)$$

where I_t is an isotonic regression (Zadrozny and Elkan, 2002) fitted entirely within the training data. We obtain its input scores through three-fold cross-validation, keeping all steps of each instance together. Each score therefore comes from classifiers trained without that instance. Evaluation instances are excluded from both classifier fitting and calibration. With forward targets, the tier-requirement classifier predicts sufficiency for the remaining trajectory, while the other two classifiers assess local feasibility and the absence of a near-term burst. Calibration targets $R_k \leq t$ with forward targets and $r_k \leq t$ with benchmark labels. Thus the two versions use probabilities of different events to make the same permanent-handoff decision.

A tier is eligible when $p_t \geq p_{\min}$. Among eligible tiers, the router maximises the handoff score

$$V_t = p_t S_t - (1 - p_t) F_t, \quad (4)$$

where S_t forecasts the saving from a successful handoff and F_t assigns a cost penalty to failure. The router hands off when the largest score is positive, preferring the cheaper tier on ties. Otherwise, it retains tier 3 and reconsiders at the next step.

Static evaluation uses price-based cost proxies. The saving S_t uses the input-price difference and forecast remaining prompt volume. The penalty

F_t uses the strongest tier’s input price and forecast whole-instance prompt volume, scaled by a weight β . Inner cross-validation selects $p_{\min}$ and β , see Appendix A. For Table 5, we train one version on benchmark labels and one on forward targets. They differ only in the targets of the tier-requirement and burst classifiers and in the calibration event, and both are scored on the same five folds that hold out whole instances.

For live evaluation, we refit the forward-target router on the full static bank and use a fixed floor of $p_{\min} = 0.95$, live cost forecasts, and a spending and step guard. Appendix A specifies these settings.

SR-KNN baseline. As a reference, we use SR-KNN, the nearest-neighbour (Cover and Hart, 1967) baseline of semantic routing (Liu et al., 2026). We use a held-out, five-neighbour version, whereas TwinRouterBench reports SR-KNN as an in-sample one-neighbour upper bound (Yang et al., 2026). On our dataset, it averages the measured success of the five most similar training cells of the same ordered model pair, weighted by similarity, and uses the decision rule of Section 3.4. On TwinRouterBench, it compares the mean tier label of the five most similar training steps with a threshold selected by cross-validation inside the training folds. It makes this comparison at the first step and keeps the chosen tier, the cheapest or the strongest, for the whole instance.

3.4 Initial Selection and Later Handoff

On our three datasets, we compare initial selection and later handoff using the same fitted router: the direct router, the factored router, or SR-KNN. At $k = 0$, before any model has acted, both policies choose among the three models. Initial selection keeps that choice for the entire task. Later handoff also commits immediately if the choice replaces the designated starting model. If the choice retains the starting model, it may reconsider at later collected steps that the trajectory reaches, before the median task length in the training data, see Appendix C.

At a later decision, a handoff occurs only when another model has strictly greater predicted utility than staying. The receiving model then finishes the task. If no handoff occurs, the starting model finishes it. The rule estimates no separate value for waiting. Both policies share the fitted predictor, opening, cost forecasts, and tie rule, with staying preferred on exact ties.

We evaluate each policy by mean task success Q and mean billed execution cost C . Each task contributes the selected cell’s average outcome over its two executions, with router initialisation seeds averaged within task. Utility is

$$U_\lambda = Q - \lambda_{\text{abs}} C, \quad (5)$$

where Q is a fraction and $\lambda_{\text{abs}} = \lambda / \bar{C}_a$, with $\bar{C}_a$ the starting model’s mean training solo cost, computed within each fold. Costs in this expression are in dollars per task, and tables report 10^{-3} \$ per task. Router computation is excluded. At $\lambda = 1$, an additional cost equal to 1% of $\bar{C}_a$ offsets one percentage point of success. We evaluate 15 dimensionless weights from 0 to 50. Equal weights across starts or datasets need not represent equal prices in dollars.

Both policies choose by predicted utility. At history h_k , a candidate model b receives

$$\hat{U}_\lambda(b \mid h_k, a) = \hat{q}(b \mid h_k, a) - \lambda_{\text{abs}} \hat{c}(a, b, k), \quad (6)$$

where $\hat{q}$ predicts terminal task success from the observed prefix, and $\hat{c}(a, b, k)$ forecasts total episode cost from training cells. For $b = a$, the score assumes that the starting model finishes the task. For another model, it assumes a handoff now followed by permanent use of that model.

We measure the paired difference, later handoff minus initial selection, in success, cost, and utility over all eligible evaluation tasks. Decisions use one observed trajectory per task and starting model, while outcomes come from the separately collected execution cell selected by the policy. This assumes that the cell’s outcome applies after the observed prefix. We assess sensitivity to this assumption by restricting evaluation to identical prefixes.

We also compare initial selection with a fixed policy: for each cost weight, the solo model or fixed handoff schedule, a handoff to a given model at a given step, with the highest mean training utility, applied to every evaluation task, see Appendix C.

Intervals resample tasks (10,000 paired resamples, seeds averaged within task) and condition on the fitted models. Primary comparisons are the direct router at the 35B start on BFCL and MultiWOZ, with simultaneous 95% bands over all weights and metrics, adjusted across the two datasets. Secondary comparisons (other starts, other routers, and the routers against the training-selected fixed policy, on BFCL and MultiWOZ) use unadjusted 95% intervals. All other comparisons

are descriptive, including τ -retail, the router comparison at initial selection, the supervision targets, and TwinRouterBench. We call a difference established when its interval excludes zero. An interval containing zero does not establish equivalence.

4 Datasets

We use two complementary resources: our dataset measures task outcomes under committed handoffs, while TwinRouterBench supplies verified benchmark labels for comparing supervision targets, see Section 5.3.

4.1 Source Datasets and Collection

We collect handoff curves on three datasets. BFCL’s `multi_turn_base` category (Patil et al., 2025) tests tool use across user turns, with decisions indexed by user turn. Its official metric checks the environment state and the execution responses after every turn. We instead count a task as solved when the model reaches the reference final state by any sequence of calls, after normalising whitespace and trailing punctuation, and we exclude tasks whose reference calls fail. τ -retail (Yao et al., 2025) pairs a retail agent with a simulated user. Its decisions are indexed by agent action, and success requires a final reward of one. MultiWOZ 2.4 (Budzianowski et al., 2018; Ye et al., 2022) provides dialogue state tracking tasks indexed by dialogue turn. A dialogue succeeds when the F1 score between its final predicted state and the gold state, computed over slot–value pairs, is at least 0.9. This measures state tracking, not general dialogue completion. On all three datasets, task success is the share of executions that succeed. Because the success criteria differ, equal success rates on two datasets do not mean equal difficulty.

Collection begins with solo screening to identify tasks on which models differ. Selected tasks receive handoff executions at the collected steps in Table 2, retaining both successful and failed outcomes. This screening enriches routing opportunities, so the resulting populations are not representative samples of the full benchmarks.

The model pool comprises Qwen3 8B (Yang et al., 2025), Qwen3.5 35B-A3B, and Qwen3.5 397B-A17B (Qwen Team, 2026), served through OpenRouter. The τ -retail user simulator is GPT-4.1-mini (OpenAI, 2025), and BFCL and MultiWOZ use scripted user turns. Dataset costs are agent-side execution costs, from provider bills ex-

Dataset	Upward	Downward	Step unit
BFCL	1–6	1–6	User turn
MultiWOZ	1–11	1–11	Dialogue turn
τ -retail	1, 2, 4, 6, 8	2, 4, 6, 8, 10	Agent action

Table 2: Collected handoff steps by dataset and direction. At step k , the starting model has completed k steps and the receiving model takes over. Initial selection at $k = 0$ uses solo executions. Evaluation considers reachable collected steps before the training-derived cutoff described in Appendix C.3.

Dataset	Cells	Executions	Evaluation tasks		
			8B	35B	397B
BFCL	2,247	4,494	102	102	102
MultiWOZ	12,504	25,008	117	117	117
τ -retail	2,351	4,702	41	17	17
Total	17,102	34,204			

Table 3: Dataset size. Cells and executions cover all tasks (training, screening, and evaluation). The last three columns give the evaluation tasks for each starting model.

cept for 431 re-collected executions priced at list price. Simulator spending is recorded separately where available and is not included in the routing cost. Appendix B gives further information.

4.2 Coverage and Splits

Each cell holds two executions, see Table 3. Cell averages provide the training labels. Tasks, not cells, are the units of evaluation.

BFCL uses task-grouped cross-validation as it does not possess a separate validation or test split. τ -retail and MultiWOZ have disjoint training and evaluation tasks. BFCL and MultiWOZ support all three starts on the same evaluation populations. In τ -retail, incomplete downward handoff coverage limits the 35B and 397B starts to 17 tasks, compared with 41 at 8B. We report those smaller populations with particular caution, i.e. all τ -retail comparisons remain descriptive rather than establishing cross-dataset effects. We will release the dataset.

4.3 TwinRouterBench

TwinRouterBench (Yang et al., 2026) is an agentic routing benchmark with two parts. Its static bank has 970 labelled steps from 520 instances, each one agent trajectory. Of these instances, 81 have more than one labelled step, and 40, the SWE subset, are software engineering tasks from SWE-bench

Verified (Jimenez et al., 2024). Each step has a verified benchmark label, from which we build our forward targets, see Section 3.2. On the bank, we hold out whole instances and score each model choice against these labels, not against new executions. The benchmark scores the routers based on four metrics. RowPass counts steps given at least the labelled tier, RowExact counts steps given exactly that tier, and TrajPass counts steps whose whole instance passes. CostSave is the share of the always-strongest bill saved under the benchmark’s cost model, which bills the whole prompt as a new cache write at every tier switch, and it can be negative because a failed instance is charged a rerun on the strongest tier. Combined is the mean of the four and is not live task success. The benchmark’s live track runs a router on 100 held-out instances. The benchmark uses a different model pool, success measure, and cost model, so its scores are not comparable with those on our dataset.

5 Results

All routers use offline full-information supervision and causal inputs, see Section 3.3. Evaluation populations are given in Table 3. Unless stated otherwise, results are at the 35B start, the only start that permits handoff in both directions. Results at the other starts are in Appendix D.

5.1 Later Handoff

No improvement over initial selection is established for the direct router at the 35B start on BFCL or MultiWOZ at any tested cost weight, see Figure 2. On BFCL, later handoff adds cost: at $\lambda \in [0, 0.35]$ it spends 0.75 to 1.6×10^{-3} \$ more per task with no established gain in success, and at $\lambda \in [0.2, 1.5]$ it lowers utility. On MultiWOZ, it spends 0.08 to 0.13×10^{-3} \$ more per task at $\lambda \in [0.05, 0.5]$, and at $\lambda \in [0.75, 1.5]$, where only 7 to 17% of dialogues succeed, it saves 0.05 to 0.07×10^{-3} \$ per task but loses 5.3 to 7.7 points of success. No utility difference is established at any cost weight on MultiWOZ. Appendix D reports sensitivity checks for fold splits, tie rules, cost forecasts, prefix identity, and waiting rules.

The losses of later handoff come from handoffs that the router ranks poorly. After the first step, the rank correlation between its predicted and realised success gain from handing off is about 0.07 on both datasets, where 0 means no rank association. On BFCL at $\lambda = 0.2$, the router hands off later to 8B

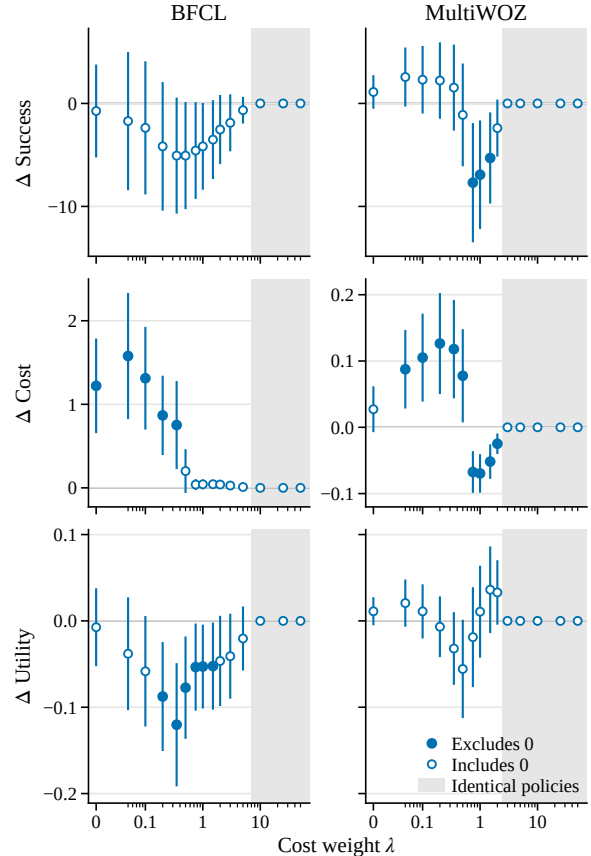


Figure 2: Effect of allowing later handoff for the direct router at the 35B start. Each panel shows later handoff minus initial selection, so values above zero mean higher success, cost, or utility with later handoff. Bars are simultaneous bands that cover all cost weights and metrics at once, at 97.5% per dataset and 95% over both datasets, and filled markers show bands that exclude 0. Success differences are in percentage points and cost differences in 10^{-3} \$ per task.

98 times across its six training seeds, on 49 tasks, predicting higher success than staying despite a higher forecast cost. These handoffs succeed 44.4% of the time at 4.75×10^{-3} \$ per task, whereas staying on 35B would have succeeded 77.0% of the time at 4.47×10^{-3} \$. On MultiWOZ, the router systematically under-predicts success from staying. With such a weak ranking, the comparison has little power to detect value in timing.

The factored router and SR-KNN behave in the same way: at $\lambda = 0.2$ on BFCL, their later handoffs cost 0.78 and 0.57×10^{-3} \$ more per task and lower utility, with no established change in success, see Table 6. The 8B and 397B starts permit only upward and downward handoffs, respectively, see Figure 4. At the 8B start on BFCL, later handoff adds 4.7 points of success at $\lambda = 1$ at almost no

extra cost, which we state as a hypothesis for a new test. On the 47 tasks where it hands off later, an immediate upward handoff, at $k = 0$, would have done better, with 72.1% against 32.1% success for 3.83 against 3.31×10^{-3} \$ per task. At the 397B start and $\lambda = 0.2$, later handoffs lower both success and cost on both datasets, see Table 6.

5.2 Router Comparison

Because later handoff adds no established benefit, we compare the routers under initial selection, see Figure 3. By unadjusted 95% intervals of the paired utility difference, the factored router has higher utility than the direct router on MultiWOZ at 6 of the 15 cost weights, with no established decrease at any weight. On BFCL, the factored router has higher utility than the direct router at only 2 of the 15 weights. On MultiWOZ, SR-KNN has higher utility than the factored router at 3 of the 15 weights and lower utility at none, and its highest success is also higher, 86.8% against 83.8%. On BFCL and τ -retail, the factored router reaches the higher success, 86.4% against 82.4% and 59.4% against 41.2%, at higher cost, 11.69 against 10.36 and 27.64 against 14.36×10^{-3} \$ per task. A likely reason for the factored router’s gain is its training target: it learns from success curves made monotone across handoff steps, the same projection that improves the direct router in Section 5.3. On τ -retail, which has only 17 evaluation tasks at this start, the comparison is descriptive.

At one low cost weight, $\lambda = 0.05$, the factored router has higher utility than the training-selected fixed policy on BFCL, by 0.055 [0.015, 0.098], with 8.5 points more success for 2.30×10^{-3} \$ more per task. At $\lambda = 0$, both the factored and the direct router reach slightly higher success than always using 397B at 17 to 23% lower cost, although the success intervals include zero. At $\lambda \in [0.2, 2]$ on BFCL, the fixed policy, which stays on 35B, beats the factored router, and at $\lambda \in [0.2, 3]$ it beats SR-KNN. On MultiWOZ, the factored router establishes no utility improvement over the fixed policy at any weight.

5.3 Supervision Targets

On our dataset, projected labels improve the direct router under initial selection, see Table 4. At the same λ , the two label types can reach different costs, so we compare each projected operating point with the success that raw labels reach at the same cost. At the same cost, projected labels reach

0.2 to 4.1 points more success in all eight cases, with the largest gain on BFCL at $\lambda = 0.5$. Utility, which accounts for cost at each weight, rises in 5 of the 8 cases, with no established decrease, see Table 8 for all 15 weights. On MultiWOZ, most of the gain at the same weight comes from a more expensive operating point. Nevertheless, projected labels improve the router consistently, with more success at the same cost in all eight cases and no established decrease in utility at any tested weight. Projected labels also raise later handoff, by 2.5 points of success on BFCL and 11.5 on MultiWOZ at $\lambda = 0.2$, for 0.18 and 0.21×10^{-3} \$ more per task. Later handoff therefore still adds no established benefit. On BFCL it lowers utility by 0.100 [0.060, 0.141] under projected labels, and on MultiWOZ no utility difference is established.

On TwinRouterBench, our forward targets improve the burst-aware router over the benchmark labels, see Table 5. RowExact rewards matching the benchmark’s tier label at each step, whereas CostSave also accounts for the cost of changing tiers, see Section 4.3. Following labels that rise and fall can therefore improve RowExact while incurring additional cache-write costs. Both burst-aware variants use the same one-handoff rule. Forward targets change when and to which tier the router hands off, but they do not introduce the one-handoff restriction. We report all four components alongside Combined to distinguish tier matching, successful execution under the benchmark labels, and cost saving.

On the full bank, forward targets raise RowPass from 95.1 to 99.3, TrajPass from 90.3 to 98.1, and CostSave by 6.3 points. On SWE, the router fails 1 instance instead of 10, with TrajPass higher by 20.5 points and CostSave by 3.3 points, although its 95% interval, $[-4.1, 8.3]$, includes zero. SR-KNN’s higher full-bank Combined comes only from RowExact, and on the full bank the forward-target router is higher on RowPass, TrajPass, and CostSave. The comparison changes three training targets together, not one label. On the live track, the router trained with forward targets resolves 79 of 100 instances at a router cost of \$0.59 per instance, see Appendix D.3. On both our dataset and TwinRouterBench, changing only the supervision targets, with the learner, features, and decision rule fixed, improves the trade-off between success and cost.

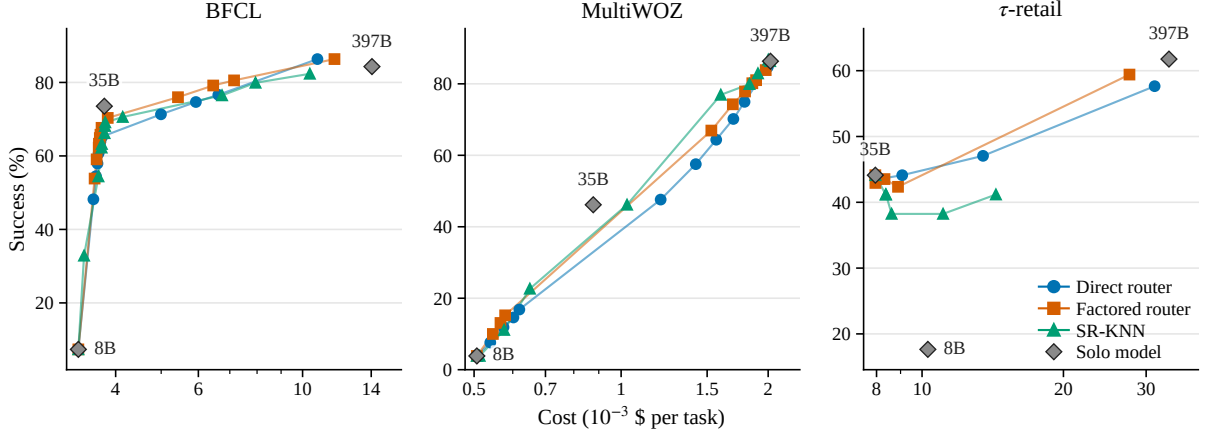


Figure 3: Initial selection at the 35B start. Each marker is a router’s operating point at one of the 15 cost weights, and lines join the points in order of increasing λ . τ -retail has 17 evaluation tasks at this start.

λ	Projected labels		Raw labels		Projected — Raw		
	Success (%)	Cost (10^{-3} \$)	Success (%)	Cost (10^{-3} \$)	Δ Success (%)	Δ Utility	Δ Success at same cost (%)
BFCL							
0	86.6	12.59	86.4	10.75	0.2 [−2.6, 3.3]	0.002 [−0.026, 0.033]	0.2
0.2	73.7	4.93	71.3	4.99	2.4 [−0.1, 5.0]	0.027 [0.003, 0.051]	2.7
0.5	69.0	3.74	64.0	3.71	5.1 [2.3, 8.0]	0.047 [0.021, 0.075]	4.1
1	64.9	3.70	62.1	3.70	2.8 [0.5, 5.1]	0.028 [0.008, 0.050]	2.9
MultiWOZ							
0	86.3	2.02	84.7	2.00	1.6 [0.6, 2.8]	0.016 [0.006, 0.028]	1.6
0.2	75.4	1.75	64.4	1.57	11.0 [7.0, 15.1]	0.067 [0.038, 0.097]	2.2
0.5	56.5	1.36	47.6	1.20	8.9 [4.4, 13.5]	0.001 [−0.033, 0.036]	1.9
1	15.4	0.60	14.6	0.60	0.8 [−1.4, 3.0]	0.007 [−0.011, 0.027]	0.7

Table 4: Direct router at the 35B start under initial selection, trained on projected or raw labels, see Section 3.2, and scored on the same raw outcomes. Success in %, success differences in percentage points, cost in 10^{-3} \$ per task. Δ Success and Δ Utility compare the two label types at the same λ , with unadjusted 95% intervals, and bold utility differences have intervals that exclude zero. Δ Success at same cost compares each projected operating point with the Pareto front of the raw-label operating points over all 15 weights. Linear interpolation between front points gives the raw-label success at the projected point’s cost, and beyond the most expensive front point the front keeps its highest success.

6 Discussion

Initial selection. At the 35B start, allowing a later handoff adds no established benefit for the direct router, the factored router, or SR-KNN, and on BFCL it adds cost. The routers’ results are therefore decided mostly by the model they select at the first step. This result depends on our model pool and on our costs, which are mostly provider bills that include prompt-cache discounts, while a handoff to another model starts without that cache. The Qwen experiments do not test transfer across model families. The result also reflects routers whose ranking of handing off against staying is close to chance, so it is not evidence that timing has no value. Because our dataset measures the

outcome of a handoff at every collected step, better predictors can be tested on it directly.

Fixed policies. A fixed policy chosen on training data is hard to beat, and how much a router can gain depends on the model pool. On BFCL, 35B solves 73.5% of tasks at 3.78×10^{-3} \$ per task, against 84.3% at 14.05×10^{-3} \$ for 397B. At $\lambda = 0$, both routers reach slightly higher success than always using 397B at 17 to 23% lower cost, although the success intervals include zero, and at $\lambda = 0.05$ the factored router also has higher utility. On MultiWOZ, 35B solves 46.2% of dialogues at 0.88×10^{-3} \$ per task, against 86.3% at 2.02×10^{-3} \$ for 397B, and the routers stay close to 397B in both success and cost. In all three

Policy	RowPass	RowExact	TrajPass	CostSave	Combined
<i>Full bank</i>					
Always strongest	100.0	17.5	100.0	0.0	54.4
Burst-aware, benchmark labels	95.1	20.1	90.3	54.3	64.9
Burst-aware, forward targets	99.3	<u>26.1</u>	98.1	60.6	<u>71.0</u>
SR-KNN	<u>95.7</u>	76.8	<u>93.0</u>	<u>55.4</u>	80.2
<i>SWE</i>					
Always strongest	100.0	50.0	100.0	0.0	62.5
Burst-aware, benchmark labels	88.4	41.4	76.8	−1.2	51.3
Burst-aware, forward targets	99.7	50.6	<u>97.3</u>	2.1	62.4
SR-KNN	<u>98.2</u>	<u>49.7</u>	97.6	<u>−0.1</u>	<u>61.4</u>

Table 5: TwinRouterBench static scores (%). Our burst-aware router is trained on the benchmark labels or on our forward targets, see Section 3.2. Benchmark labels are TwinRouterBench’s tier labels for each step, and forward targets give each step the strongest tier needed from that step to the end of the instance. The full bank has 520 instances with 970 labelled steps, and SWE is its subset of 40 SWE-bench Verified instances. Always strongest, in grey, is a reference. Among the three router rows, the highest value in each column is in bold and the second highest is underlined.

datasets, 8B fails most tasks, so routing to it saves little. This suggests that a pool whose models are closer in capability but differ in price would leave more room for routing to improve the trade-off between success and cost. Router evaluations should report the training-selected fixed policy next to the router.

Supervision targets. The clearest gains in our study come from what the routers learn to predict, not from when they may hand off. On our dataset, making training curves monotone gives the direct router 0.2 to 4.1 points more success at the same cost in all eight reported cases, and raises its utility at 7 of the 15 cost weights on BFCL and at 4 on MultiWOZ, with no decrease established at any tested weight. The factored router, which learns from monotone curves, has higher utility than the direct router at 6 of the 15 weights on MultiWOZ. On TwinRouterBench, forward targets raise the burst-aware router’s TrajPass by 7.8 points and its CostSave by 6.3 points on the full static bank, and on SWE they reduce failed instances from 10 to 1. A router trained with these targets resolves 79 of 100 live instances at a router cost of \$0.59 per instance. On our dataset, projected labels improve both policies by similar amounts.

References

- Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. 2024. AutoMix: Automatically mixing language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 131000–131034.
- Richard E. Barlow, David J. Bartholomew, J. M. Bremner, and H. D. Brunk. 1972. *Statistical Inference Under Order Restrictions*. Wiley.
- Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. [MultiWOZ – a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5016–5026.
- Josef Chen. 2026a. When does combining language models help? A co-failure ceiling on routing, voting, and mixture-of-agents across 67 frontier models. arXiv preprint arXiv:2606.27288.
- Lingjiao Chen, Matei Zaharia, and James Zou. 2023. FrugalGPT: How to use large language models while reducing cost and improving performance. arXiv preprint arXiv:2305.05176.
- Teng-Ruei Chen. 2026b. What does a routing oracle measure under stochastic decoding? Coupling, scorer choice, and single-commit ceilings. arXiv preprint arXiv:2607.03436.
- Thomas M. Cover and Peter E. Hart. 1967. [Nearest neighbor pattern classification](#). *IEEE Transactions on Information Theory*, 13(1):21–27.
- Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed H Awadallah. 2024. Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*.
- Bradley Efron. 1979. [Bootstrap methods: Another look at the jackknife](#). *The Annals of Statistics*, 7(1):1–26.

- Jerome H. Friedman. 2001. [Greedy function approximation: A gradient boosting machine](#). *The Annals of Statistics*, 29(5):1189–1232.
- Roy Ganz, Mor Shpigel Nacson, Adi Kalyanpur, and Ron Litman. 2026. The handoff tax: Continuing non-native trajectories in LLM agents. *arXiv preprint arXiv:2608.24358*.
- Ashritha Gonuguntla. 2026. The replay gap: Static evaluation of model switching in LLM agents scores the wrong world. *arXiv preprint arXiv:2608.08239*.
- Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Router-Bench: A benchmark for multi-LLM routing system. *arXiv preprint arXiv:2403.12031*.
- Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, volume 2024, pages 54107–54157.
- Vansh Kapoor, Aman Gupta, Hao Chen, Anurag Beniwal, Jing Huang, and Aviral Kumar. 2026. TRIM: Hybrid inference via targeted stepwise routing in multi-step reasoning tasks. *arXiv preprint arXiv:2601.10245*.
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30.
- Raad Khraishi, Iman Zafar, Katie Myles, and Greig A Cowan. 2026. Evaluating performance drift from model switching in multi-turn LLM systems. *arXiv preprint arXiv:2603.03111*.
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*. *ArXiv:1412.6980*.
- Janghoon Lee. 2026. Most of the LLM routing gap is task type. *arXiv preprint arXiv:2608.23023*.
- Ji’an Lei and Jian Huang. 2026. TACIT-Switch: Cost-aware model escalation for LLM agents from censored supervision. *arXiv preprint arXiv:2608.27911*.
- Yuanzhe Li, Jianing Deng, Jingtong Hu, Tianlong Chen, Song Wang, and Huanrui Yang. 2026. Dynamic mixed-precision routing for efficient multi-step LLM interaction. *arXiv preprint arXiv:2602.02711*.
- Xunzhuo Liu, Huamin Chen, Samzong Lu, Yossi Ovardia, Guohong Wen, Hao Wu, Zhengda Tan, Jintao Zhang, Senan Zedan, Yehudit Kerido, et al. 2026. vLLM semantic router: Signal driven decision routing for mixture-of-modality models. *arXiv preprint arXiv:2603.04444*.
- José Luis Montiel Olea and Mikkel Plagborg-Møller. 2019. [Simultaneous confidence bands: Theory, implementation, and an application to SVARs](#). *Journal of Applied Econometrics*, 34(1):1–17.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M. Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations*.
- OpenAI. 2025. Introducing GPT-4.1 in the API. <https://openai.com/index/gpt-4-1/>.
- Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.
- Dzianis Piatrashyn, Nikita Kotelevskii, Kirill Grishchenkov, Nikita Glazkov, Ivan Nasonov, Ilya Makarov, Timothy Baldwin, Preslav Nakov, Roman Vashurin, and Maxim Panov. 2026. ReDAct: Uncertainty-aware deferral for LLM agents. *arXiv preprint arXiv:2604.07036*.
- Qwen Team. 2026. Qwen3.5: Towards native multi-modal agents. <https://qwen.ai/blog?id=qwen3.5>.
- Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992.
- Ibne Farabi Shihab, Abu Sa-Adat Mohamed Moon-Im Al Ahsan, and Md Najmus Swaqeeb. 2026. Opportunity is not realizability: Selection-valid diagnostics for multi-LLM routing. *arXiv preprint arXiv:2608.08265*.
- Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2024. Large language model routing with benchmark datasets. In *First Conference on Language Modeling (COLM)*. *ArXiv:2309.15789*.
- Seongho Son, Sangwoong Yoon, Jiahua Tang, Shuhan Wang, Lorenz Wolf, and Ilija Bogunovic. 2026. SWE-Router: Routing in multi-turn agentic software engineering tasks. *arXiv preprint arXiv:2607.00053*.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788.

Zhongling Xu, Shunan Zheng, and Wei Wang. 2026. SeqRoute: Global budget-aware sequential LLM routing via offline reinforcement learning. arXiv preprint arXiv:2605.25424.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. 2025. Qwen3 technical report. arXiv preprint arXiv:2505.09388.

Pei Yang, Wanyi Chen, Tongyun Yang, Pengbin Feng, Jiarong Xing, Wentao Guo, Yuhang Yao, Yuhang Han, Hanchen Li, Xu Wang, et al. 2026. TwinRouterBench: Fast static and live dynamic evaluation for realistic agentic LLM routing. arXiv preprint arXiv:2605.18859.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2025. τ -bench: A benchmark for tool-agent-user interaction in real-world domains. In International Conference on Learning Representations (ICLR). ArXiv:2406.12045.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2022. MultiWOZ 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. In Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 351–360.

Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 694–699.

Chubin Zhang, Zhenglin Wan, Xingrui Yu, Jingxuan Wu, Qi Wen, Pengfei Zhou, Wangbo Zhao, and Ivor Tsang. 2026a. Calibration is not control: Intervention value for LLM-agent oversight. arXiv preprint arXiv:2606.21399. NeurIPS 2026.

Jiarui Zhang, Xiangyu Liu, Yong Hu, Chaoyue Niu, Hang Zeng, Shaojie Tang, Fan Wu, and Guihai Chen. 2026b. From myopic selection to long-horizon awareness: Sequential LLM routing for multi-turn dialogue. arXiv preprint arXiv:2604.12385.

Yiqun Zhang, Hao Li, Zihan Wang, Shi Feng, Xiaocui Yang, Daling Wang, Bo Zhang, Lei Bai, and Shuyue Hu. 2026c. MTRouter: Cost-aware multi-turn LLM routing with history–model joint embeddings. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 44206–44226.

A Prediction and Live Policy Details

A.1 Dataset Router Training

The direct router is fitted once per dataset, fold, and initialisation seed, pooling all three starting models. In its weighted cross-entropy loss, when a cell supplies several views, including solo anchors and stay rows at several handoff steps, those views share

its total weight of two within each fitted objective. Zero-success cells remain in raw-label training.

All routers use frozen all-MiniLM-L6-v2 embeddings with normalised 384-dimensional outputs (Wang et al., 2020; Reimers and Gurevych, 2019). The direct router concatenates the instruction embedding of the first 2,000 characters, 11 history statistics, 32 history-embedding PCA components fitted on training rows, and 11 model, step, and price features. History statistics describe actions, results, errors, history length, user turns, and the size and parse health of the carried state. The static features encode the model pair, direction, current and candidate steps, their difference, and a price proxy at the median training task length. In all reported analyses, the candidate step is the current step. At step zero, the start, direction, and price features are zeroed.

Neural heads use standard scaling, ReLU activations, and Adam (Kingma and Ba, 2015), with learning rate 10^{-3} , L_2 coefficient 10^{-4} , and at most 700 iterations. BFCL uses partition seed 0, and seeds 1 and 2 test sensitivity to the split.

For the direct router’s label comparison, projection is fitted separately by training task and ordered model pair. It includes the receiving-model solo anchor and leaves stay labels raw. Zero-anchor curves remain in training without an anchor cap. All other training and evaluation settings are held fixed.

A.2 Factored Router and SR-KNN Baseline

For the factorisation in Section 3.3, curves with $d \leq 10^{-6}$ are excluded from fitting the timing head g but remain in fitting the difficulty head d and in evaluation. Reconstructed training targets are capped at the solo anchor. The direct router’s projected labels have no such cap. Both heads use the same neural settings as above. The difficulty head sees the instruction and the stronger-model indicator on BFCL, or the receiving-model indicator within each fixed start on MultiWOZ and τ -retail. The timing head sees the instruction, 11 history statistics, starting- and receiving-model indicators, and handoff step ($k/16$ on BFCL and τ -retail, $k/12$ on MultiWOZ). Predicted timing profiles are projected using only steps observed so far, with the upward handoff profile anchored at one at step zero.

SR-KNN draws neighbours from the same ordered model pair, with one pool entry per training task and measured cell. Similarity is the dot product of concatenated normalised instruction and

history embeddings. SR-KNN averages the raw labels of the five most similar cells, with similarity weights floored at 10^{-6} .

A.3 Bank Relabelling and Live Handoff

TwinRouterBench verifies each tier substitution under its search procedure (Yang et al., 2026). Its step labels do not measure the outcome of a permanent handoff, see Section 3.2.

The classifiers use scikit-learn’s histogram gradient boosting (Friedman, 2001; Ke et al., 2017), with seed 0, learning rate 0.1, and at most 100 iterations and 31 leaves. History excludes system messages, limits messages to 300 characters and tool-call arguments to 160, and retains the final 4,000 characters. Role-delimited segments are limited to 1,500 characters, with tool outputs attached to the preceding segment. Features concatenate the embedding of the first 2,000 characters of the first user message, the embeddings of the last four segments with zero left-padding, the mean of all segment embeddings, and a scenario indicator. With forward targets, the burst classifier has three classes: a burst at the current step, a burst within the next three labelled steps, or neither. Its predicted probability of the last class is p_{clear} . With benchmark labels, it has two classes, a burst at the current step or not, and p_{clear} is the probability of no burst.

Static evaluation uses five outer folds grouped by instance. Within each training fold, three inner folds supply calibration scores and select $p_{\min} \in \{0, 0.95, 0.975, 0.98, 0.99\}$ and $\beta \in \{0.5, 1, 2\}$ by the highest Combined score among settings with positive CostSave. Evaluation uses the original benchmark labels.

Static cost forecasts use $S_t = (\rho_3 - \rho_t)\hat{W}_{\text{rem}}$ and $F_t = \beta\rho_3\hat{W}_{\text{inst}}$, where ρ_t is the tier’s input price per token, and $\hat{W}_{\text{rem}}$ and $\hat{W}_{\text{inst}}$ are the forecast prompt volumes of the remaining steps and of the whole instance. Each prompt-volume forecast multiplies the current history length by a training mean ratio for remaining or total volume, grouped by scenario and step bin: 0, 1–4, 5–14, 15–39, or 40 and above.

For live use, the router is refitted on the full static bank with three instance-grouped calibration folds, the software-engineering scenario indicator, and $p_{\min} = 0.95$. Forecasts assume an instance length of 32 steps, at least five remaining steps, 500 output tokens per step, and one prompt token per four characters, using locked input and output prices without cache discounts. S_t is the remaining

cost difference between tiers 3 and t , and F_t is the forecast tier- t spend plus the \$0.60 unresolved-instance penalty. Every eligible tier has positive V_t , so the floor $p_{\min}$ controls handoff timing and the handoff score V_t selects the tier. If no handoff has occurred, spending above \$1.50 or a step count above 70 triggers a permanent handoff to tier 2. Appendix D.3 reports the live evaluation.

B Data Collection and Provenance

BFCL’s solo inventory includes 95 screening-only tasks outside the 102 tasks with handoff curves used for evaluation.

B.1 Collection and Serving Conditions

BFCL and MultiWOZ have both handoff directions for every selected task, using the steps in Table 2. In τ -retail, all 41 evaluation tasks have upward handoff curves, but only 17 have downward handoff curves. The 28 upward and 29 downward training tasks are disjoint, leaving no training task with both directions at the 35B start. Nine downward training cells are missing: seven had only one valid execution and two were never collected. MultiWOZ training and evaluation partitions both come from the MultiWOZ 2.4 validation set.

Qwen3 8B, Qwen3.5 35B-A3B, and Qwen3.5 397B-A17B were served through OpenRouter and pinned to the providers Alibaba, Parasail, and Chutes, respectively. Provider changes, fallbacks, and campaign-specific exceptions are recorded by campaign and, where available, by call.

B.2 Replication and Exclusions

The dataset retains two executions per cell, selected by source-file order and record index. It excludes 837 earlier-protocol τ -retail training cells: 768 replayed a recorded prefix and billed only the continuation, and 69 used a shared recorded opening. These were not separately collected full episodes.

C Evaluation and Statistical Details

C.1 Decision Inputs and Canonical Trajectories

Timing features use the current step index, never its fraction of the final episode length. Other inputs follow the causal restrictions in Section 3.3.

Ties between alternative receiving models go to the lower forecast cost and then to the smaller model. A threshold that allows handoff on ties and

rules that give waiting an explicit value are descriptive sensitivities, see Appendix D, not replacements for the primary rule.

One canonical trajectory per task and starting model supplies every candidate’s prefix at a decision point. The canonical trajectory is the starting model’s solo execution with the lowest replicate identifier. If none exists, it is a rollout with the starting model’s prefix and the latest collected handoff step, breaking ties by fixed dataset keys, not outcomes. The trajectory is selected once, never stitched or replaced, and its termination is respected.

C.2 Cost Forecasts and Cost Weights

Cost forecasts use training cells only. Initial selection uses each model’s mean solo cost. Later handoffs use means by starting model, receiving model, and step. Staying uses solo means over training tasks whose canonical trajectory reaches that step.

Later-step means require five distinct training tasks. Unsupported steps use interpolation within the supported range, the nearest supported step outside it, or the pooled ordered-pair mean if no step is supported. Initial solo means are exempt from this requirement. The primary forecasts have no monotonicity constraint. Reported costs are the selected cells’ billed execution costs.

The 15 cost weights are 0, 0.05, 0.1, 0.2, 0.35, 0.5, 0.75, 1, 1.5, 2, 3, 5, 10, 25, and 50. The normaliser $\bar{C}_a$ is computed for each fold, and at the 35B start it is 3.78×10^{-3} \$ per task on BFCL, averaged over folds, and 0.87×10^{-3} \$ on MultiWOZ. No cost weight is selected on evaluation outcomes.

C.3 Evaluation Populations and Uncertainty

Evaluation tasks are excluded from training at every starting model. Handoffs are considered only before the median training task length: 4 steps on BFCL, 8 on MultiWOZ, and 8, 10, and 12 on τ -retail at the 8B, 35B, and 397B starts. This removes 36 of 276 BFCL and 56 of 771 MultiWOZ decision points.

The data were examined before the protocol was frozen, and screening outcomes also supply the solo outcomes at $k = 0$. Evaluation therefore does not use an untouched sample, and independent solo outcomes were not re-collected. The effect of changing stay-observation weights was not tested, so the router’s under-prediction of staying on MultiWOZ remains a diagnostic rather than

an established causal explanation.

Task-bootstrap resamples (Efron, 1979) are shared across policies and cost weights. Primary simultaneous bands (Montiel Olea and Plagborg-Møller, 2019) use the maximum standardised bootstrap deviation across success, cost, utility, and all 15 weights, with error rate 0.025 per dataset for 95% coverage across BFCL and MultiWOZ. Intervals outside the primary family, including all τ -retail intervals, are unadjusted 95% percentile intervals from the same resamples.

C.4 Fixed-Policy Comparison

For each dataset, start, fold, and cost weight, the fixed policy maximises mean training utility over the three solos and handoff schedules observed on at least 80% of training tasks. At the τ -retail 35B start, only solos qualify. Ties favour lower mean billed cost, earlier handoff, and then the smaller model. Evaluation outcomes are never imputed.

For comparisons with the fixed policy, initial selection uses a standardised opening, taken from one fixed execution of each task, and solo-mean cost forecasts. On BFCL and MultiWOZ, this produces the same direct-router initial choices as the later-handoff comparison. On τ -retail, the two can differ.

D Additional Results and Sensitivities

D.1 Policy and Supervision Comparisons

Figure 4 shows initial selection and later handoff for the two routers and SR-KNN at every starting model, and Table 6 gives their paired differences at $\lambda = 0.2$. At the 35B and 397B starts, τ -retail has 17 evaluation tasks. Table 7 gives the values plotted in Figure 2. Table 8 gives the raw-against-projected label comparison of Table 4 at every cost weight.

D.2 Sensitivity Checks

At BFCL’s 35B start, waiting variants improve on strict later handoff at some cost weights. The direct waiting rule establishes no improvement over initial selection, and it adds supervision for future handoff schedules, so that comparison does not isolate the effect of waiting. The factored router’s waiting rule is compared only with strict later handoff.

BFCL partition seeds 1 and 2 preserve the signs of the checked contrasts on the same 102 tasks. Allowing handoff on ties changes no direct- or factored-router decisions, and one SR-KNN decision per seed on MultiWOZ at the 397B start.

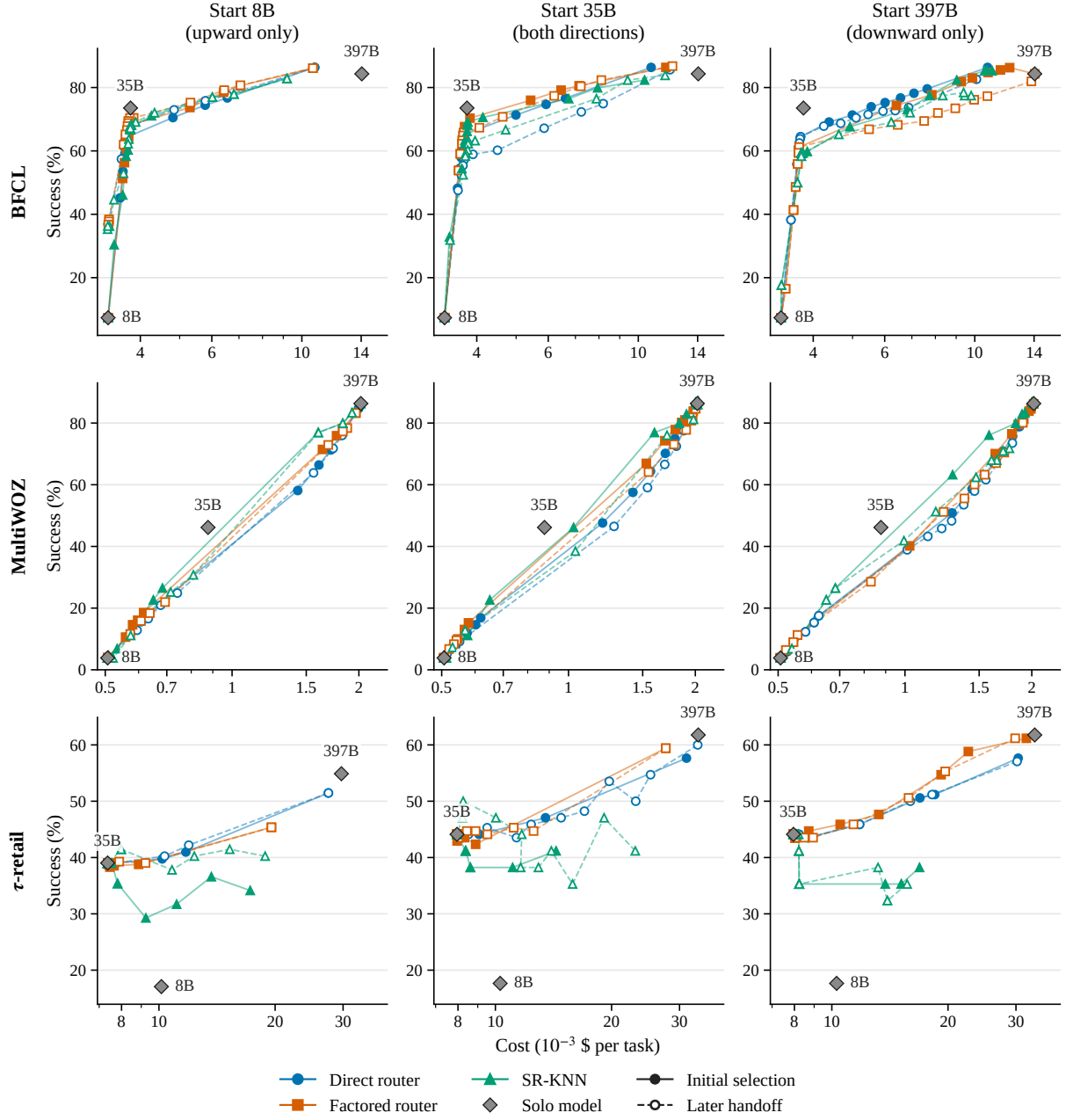


Figure 4: Initial selection (solid, filled) and later handoff (dashed, open) at every starting model over the 15 cost weights. τ -retail has 17 evaluation tasks at the 35B and 397B starts and 41 at the 8B start, see Table 3.

Start	Router	Δ Success (%)	Δ Cost (10^{-3} \$)	Δ Utility
<i>BFCL</i>				
8B	Direct	2.5 [1.1, 4.0]	0.03 [−0.01, 0.07]	0.023 [0.009, 0.038]
	Factored	1.6 [0.4, 3.1]	0.02 [−0.01, 0.06]	0.015 [0.002, 0.031]
	SR-KNN	1.0 [0.0, 2.9]	0.08 [−0.01, 0.22]	0.005 [−0.002, 0.018]
35B	Direct	−4.2 [−8.0, −0.7]	0.87 [0.60, 1.16]	−0.088 [−0.127, −0.052]
	Factored	1.4 [−0.6, 3.6]	0.78 [0.46, 1.15]	−0.027 [−0.047, −0.009]
	SR-KNN	−3.9 [−10.3, 2.0]	0.57 [0.25, 0.95]	−0.069 [−0.130, −0.014]
397B	Direct	−4.2 [−6.3, −2.2]	−0.62 [−0.82, −0.44]	−0.033 [−0.052, −0.015]
	Factored	−11.3 [−15.8, −6.9]	−1.83 [−2.25, −1.44]	−0.087 [−0.128, −0.046]
	SR-KNN	−4.9 [−9.8, −0.5]	−0.69 [−1.10, −0.32]	−0.039 [−0.087, 0.004]
<i>MultiWOZ</i>				
8B	Direct	5.7 [3.5, 8.2]	0.13 [0.09, 0.17]	0.006 [−0.014, 0.028]
	Factored	1.5 [0.3, 2.9]	0.05 [0.03, 0.08]	−0.006 [−0.018, 0.004]
	SR-KNN	0.0 [0.0, 0.0]	0.00 [0.00, 0.01]	−0.002 [−0.005, 0.000]
35B	Direct	2.2 [0.1, 4.5]	0.13 [0.08, 0.17]	−0.007 [−0.028, 0.014]
	Factored	−0.1 [−1.8, 1.6]	0.11 [0.06, 0.17]	−0.026 [−0.046, −0.008]
	SR-KNN	1.3 [−1.7, 4.3]	0.15 [0.07, 0.25]	−0.021 [−0.063, 0.014]
397B	Direct	−17.4 [−22.5, −12.5]	−0.33 [−0.40, −0.27]	−0.140 [−0.188, −0.093]
	Factored	−18.0 [−24.2, −11.9]	−0.37 [−0.44, −0.30]	−0.143 [−0.202, −0.084]
	SR-KNN	−15.4 [−21.8, −9.4]	−0.27 [−0.36, −0.19]	−0.126 [−0.187, −0.069]
<i>τ-retail</i>				
8B	Direct	0.0 [0.0, 0.0]	0.03 [0.00, 0.10]	−0.001 [−0.003, 0.000]
	Factored	0.7 [0.0, 2.0]	0.04 [−0.05, 0.16]	0.006 [−0.003, 0.019]
	SR-KNN	8.5 [1.2, 17.1]	1.56 [0.04, 3.44]	0.041 [−0.040, 0.135]
35B	Direct	10.0 [−0.6, 21.2]	11.63 [7.25, 16.41]	−0.300 [−0.443, −0.171]
	Factored	1.2 [−3.5, 7.1]	1.58 [0.40, 2.99]	−0.043 [−0.114, 0.020]
	SR-KNN	2.9 [0.0, 8.8]	5.35 [1.76, 9.48]	−0.155 [−0.295, −0.039]
397B	Direct	0.0 [0.0, 0.0]	0.00 [0.00, 0.00]	0.000 [0.000, 0.000]
	Factored	−1.8 [−3.5, 0.0]	−1.86 [−4.48, −0.20]	−0.006 [−0.027, 0.015]
	SR-KNN	2.9 [0.0, 8.8]	−0.58 [−1.73, 0.00]	0.033 [0.000, 0.099]

Table 6: Later handoff minus initial selection for each router at $\lambda = 0.2$, with success differences in percentage points and cost differences in 10^{-3} \$ per task. Intervals are unadjusted 95%, and rows of zeros mean that the two policies make identical choices. τ -retail has 41 evaluation tasks at the 8B start and 17 at the 35B and 397B starts, see Table 3.

λ	Δ Success (%)	Δ Cost (10^{-3} \$)	Δ Utility
<i>BFCL</i>			
0	-0.7 [-5.2, 3.8]	1.22 [0.65, 1.79]	-0.007 [-0.052, 0.038]
0.05	-1.7 [-8.4, 5.0]	1.58 [0.82, 2.33]	-0.038 [-0.103, 0.027]
0.1	-2.4 [-8.8, 4.1]	1.31 [0.70, 1.93]	-0.058 [-0.122, 0.006]
0.2	-4.2 [-10.4, 2.1]	0.87 [0.39, 1.34]	-0.088 [-0.151, -0.024]
0.35	-5.1 [-10.7, 0.6]	0.75 [0.22, 1.28]	-0.120 [-0.192, -0.049]
0.5	-5.1 [-10.3, 0.1]	0.20 [-0.06, 0.46]	-0.077 [-0.137, -0.018]
0.75	-4.6 [-9.3, 0.1]	0.04 [-0.03, 0.10]	-0.054 [-0.104, -0.003]
1	-4.2 [-8.4, 0.0]	0.04 [-0.02, 0.10]	-0.053 [-0.101, -0.004]
1.5	-3.5 [-7.3, 0.3]	0.04 [-0.01, 0.10]	-0.052 [-0.103, -0.002]
2	-2.5 [-5.9, 0.8]	0.04 [-0.01, 0.09]	-0.046 [-0.099, 0.006]
3	-1.9 [-4.6, 0.9]	0.03 [-0.01, 0.06]	-0.041 [-0.090, 0.008]
5	-0.7 [-1.9, 0.6]	0.01 [-0.01, 0.03]	-0.020 [-0.057, 0.017]
10-50	identical policies (all differences 0)		
<i>MultiWOZ</i>			
0	1.1 [-0.5, 2.7]	0.03 [-0.01, 0.06]	0.011 [-0.005, 0.027]
0.05	2.6 [-0.3, 5.4]	0.09 [0.03, 0.15]	0.021 [-0.007, 0.048]
0.1	2.3 [-1.0, 5.6]	0.11 [0.04, 0.17]	0.011 [-0.020, 0.042]
0.2	2.2 [-1.5, 5.9]	0.13 [0.05, 0.20]	-0.007 [-0.042, 0.028]
0.35	1.5 [-2.6, 5.7]	0.12 [0.04, 0.19]	-0.032 [-0.074, 0.010]
0.5	-1.1 [-6.1, 3.9]	0.08 [0.01, 0.15]	-0.056 [-0.113, 0.001]
0.75	-7.7 [-13.5, -1.9]	-0.07 [-0.10, -0.04]	-0.019 [-0.077, 0.039]
1	-6.9 [-12.2, -1.6]	-0.07 [-0.10, -0.04]	0.011 [-0.043, 0.064]
1.5	-5.3 [-9.7, -0.9]	-0.05 [-0.08, -0.03]	0.036 [-0.014, 0.086]
2	-2.4 [-5.2, 0.4]	-0.02 [-0.04, -0.01]	0.033 [-0.004, 0.070]
3-50	identical policies (all differences 0)		

Table 7: Later handoff minus initial selection for the direct router at the 35B start, at every cost weight, with success differences in percentage points and cost differences in 10^{-3} \$ per task. Intervals are the primary simultaneous bands of Figure 2, covering all cost weights and metrics at 97.5% per dataset and 95% over both datasets.

λ	Δ Success (%)	Δ Cost (10^{-3} \$)	Δ Utility
<i>BFCL</i>			
0	0.2 [-2.6, 3.3]	1.83 [1.34, 2.36]	0.002 [-0.026, 0.033]
0.05	4.7 [2.3, 7.3]	1.68 [1.13, 2.27]	0.025 [0.002, 0.049]
0.1	4.0 [1.4, 6.7]	0.85 [0.39, 1.34]	0.017 [-0.006, 0.041]
0.2	2.4 [-0.1, 5.0]	-0.06 [-0.42, 0.27]	0.027 [0.003, 0.051]
0.35	4.4 [1.8, 7.3]	-0.01 [-0.05, 0.03]	0.045 [0.019, 0.072]
0.5	5.1 [2.3, 8.0]	0.02 [-0.01, 0.06]	0.047 [0.021, 0.075]
0.75	3.7 [1.1, 6.4]	0.02 [-0.02, 0.05]	0.033 [0.010, 0.058]
1	2.8 [0.5, 5.1]	0.00 [-0.03, 0.03]	0.028 [0.008, 0.050]
1.5	1.9 [-0.6, 4.2]	-0.01 [-0.04, 0.03]	0.021 [-0.000, 0.042]
2	2.9 [0.2, 5.4]	0.00 [-0.03, 0.04]	0.027 [0.005, 0.049]
3	2.1 [-1.4, 5.6]	0.01 [-0.03, 0.05]	0.015 [-0.023, 0.053]
5	0.0 [-3.5, 3.4]	0.00 [-0.04, 0.04]	-0.003 [-0.055, 0.050]
10-50	identical policies (all differences 0)		
<i>MultiWOZ</i>			
0	1.6 [0.6, 2.8]	0.02 [0.01, 0.04]	0.016 [0.006, 0.028]
0.05	6.7 [3.3, 10.1]	0.11 [0.05, 0.16]	0.061 [0.030, 0.092]
0.1	9.5 [5.6, 13.4]	0.14 [0.07, 0.21]	0.079 [0.045, 0.113]
0.2	11.0 [7.0, 15.1]	0.19 [0.12, 0.26]	0.067 [0.038, 0.097]
0.35	8.9 [4.6, 13.2]	0.17 [0.09, 0.24]	0.022 [-0.005, 0.049]
0.5	8.9 [4.4, 13.5]	0.15 [0.08, 0.23]	0.001 [-0.033, 0.036]
0.75	0.9 [-1.5, 3.4]	0.01 [-0.01, 0.02]	0.001 [-0.021, 0.024]
1	0.8 [-1.4, 3.0]	0.00 [-0.01, 0.01]	0.007 [-0.011, 0.027]
1.5	-0.3 [-2.4, 1.8]	0.00 [-0.01, 0.01]	-0.003 [-0.021, 0.017]
2	1.3 [-0.8, 3.7]	0.00 [-0.01, 0.02]	0.005 [-0.016, 0.025]
3-50	identical policies (all differences 0)		

Table 8: Projected minus raw labels for the direct router at the 35B start under initial selection, at every cost weight, with success differences in percentage points and cost differences in 10^{-3} \$ per task. Intervals are unadjusted 95%.

At the 35B start, alternative cost tables change no direct-router decisions on BFCL and two on MultiWOZ. A monotone cost forecast lowers the success gain of later handoff at BFCL’s 8B start at $\lambda = 1$ from 4.7 to 3.3 points. Restricting evaluation to identical prefixes leaves the BFCL utility difference at $\lambda = 0.35$ unchanged at -0.120 on 58 tasks. On 91 MultiWOZ dialogues, the small positive utility differences at $\lambda = 1.5$ and 2 shrink to $+0.008$.

D.3 Live TwinRouterBench Evaluation

On the live track’s 100 held-out instances, the burst-aware router trained with forward targets on the full static bank resolves 79 of 100 instances at \$0.591 in router cost and \$0.717 in penalty-inclusive leaderboard bill per instance. Published reference runs resolve 74 instances for always strongest and 75 for SR-KNN, at agent costs of \$0.547 and \$0.556 per instance, or bills of \$0.703 and \$0.706 with the same penalty.

The live pool maps tiers 0–3 to DeepSeek V3.2, MiniMax M2.7, Gemini 3 Flash Preview, and Claude Opus 4.6, respectively. This is the execution pool, distinct from the static bank’s tier annotations.